AD-A070 334 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 19/6

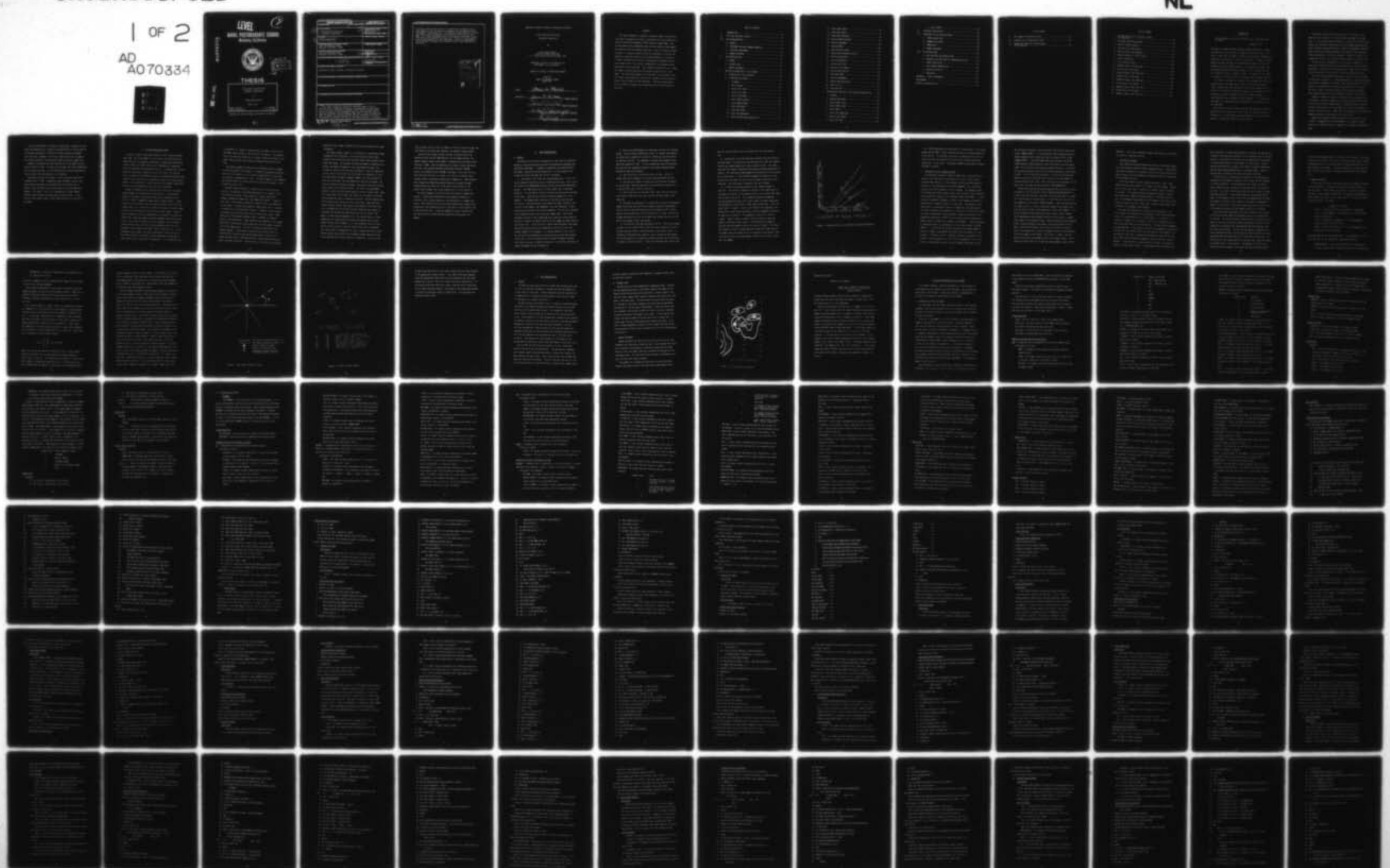
A TWO-SIDED FIELD ARTILLERY STOCHASTIC SIMULATION.(U)

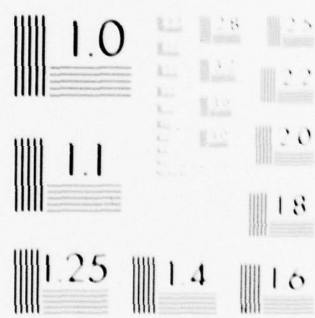
MAR 79 S G STARNER

UNCLASSIFIED NL

1 OF 2

AD
A070334





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A TWO-SIDED FIELD ARTILLERY
STOCHASTIC SIMULATION

by

Steven Gage Starner

March 1979

Thesis Advisor: J. K. Hartman
Thesis Co-Advisor: E. P. Kelleher

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) (6) A Two-Sided Field Artillery Stochastic Simulation,	5. TYPE OF REPORT & PERIOD COVERED (9) Master's Thesis, March 1979
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) Steven Gage/Starner	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	12. REPORT DATE (11) March 1979
	13. NUMBER OF PAGES 186
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) (12) 187p.	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis presents a stochastic simulation model of Field Artillery that is designed to be used in conjunction with the Simulation of Tactical Alternative Responses (STAR) battalion combat model. The tactics modeled, the assumptions made, and the interface requirements are detailed, with the computer code that is used to execute the model included. A clustering algorithm that was developed to simulate the grouping of detected vehicles by the forward observer is discussed. cont

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
(Page 1) S/N 0102-014-6601 |

79 06 20 075

251 450

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

LB

A typical simulated fire mission is presented with computer plots of each element of the fire mission, accompanied by the computer output that details the interactions. This graphical presentation will enable the reader to better appreciate the potential applications of the model. The input requirements to use the model are explained so that this thesis could become the initial user's manual for future applications of the FA model with the STAR model. The design requirements for an FA model to support the brigade version of the STAR model are discussed.

Accession For	
NTIS GRA&I	✓
DDC TAB	
Unannounced	
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or special
A	

A Two-Sided Field Artillery

Stochastic Simulation

by

Steven Gage Starner
Captain, United States Army
B.S., United States Military Academy, 1970

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
March 1979

Author: Steven G. Starner

Approved by: James K. Hartman, Thesis Advisor

Edward P. Kelleher, Thesis Co-Advisor

Chairman, Department of Operations Research

Dean of Information and Policy Sciences

ABSTRACT

This thesis presents a stochastic simulation model of Field Artillery that is designed to be used in conjunction with the Simulation of Tactical Alternative Responses (STAR) battalion combat model. The tactics modeled, the assumptions made, and the interface requirements are detailed, with the computer code that is used to execute the model included. A clustering algorithm that was developed to simulate the grouping of detected vehicles by the forward observer is discussed. A typical simulated fire mission is presented with computer plots of each element of the fire mission, accompanied by the computer output that details the interactions. This graphical presentation will enable the reader to better appreciate the potential applications of the model. The input requirements to use the model are explained so that this thesis could become the initial user's manual for future applications of the FA model with the STAR model. The design requirements for an FA model to support the brigade version of the STAR model are discussed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

"The instruments of battle are valuable only if one knows how to use them."

Ardant du Picq

The nature of combat prevents "practice" before hostilities. Peacetime exercises do not faithfully represent the actions of war. The analyst, then, must use models of combat to help develop equipment and operational techniques for the future battlefield. A number of combat models have been developed using observed and assumed relationships between the relevant combat processes. The quantitative results and the understanding gained from such models assist in determining the most effective mixture of weapons systems. The purpose of this thesis is to model the Field Artillery's contribution to the Central Battle.

This model is designed to interface with and become part of an updated version of the Simulation of Tactical Alternative Responses (STAR) combined arms model developed by Wallace and Hagewood [Ref. 1]. Written in the Simscript simulation language, the STAR model is a mid-resolution stochastic computer simulation of combat between a U.S. (Blue) Battalion and a Warsaw Pact (Red) Regiment. This Field Artillery model is a module that adds a simulation of Field Artillery to the STAR model.

The Field Artillery (FA) model considers the major Field Artillery events and actions that occur in the process of fighting the battle. Although it is a simplification and approximation of the real flow of events, it portrays the fabric of the FA battle.

The process of model building and analysis is iterative. This model builds on the logic of a previous model developed by Kelleher [Ref. 2]. The Kelleher model represented a single Blue Field Artillery battery with simplified target acquisition and damage assessment functions. The model presented here has a number of enhancements: detailed target acquisition, damage assessment, two-sided artillery, and multiple representation of artillery elements. Chapter III details the assumptions, tactics, and procedures used to model the Blue artillery.

The Red artillery representation is discussed in Chapter IV. In consonance with Warsaw Pact doctrine, the majority of the fires were preplanned. The Red artillery was modeled at a lower level of resolution than the Blue artillery due to a smaller available data base.

The logic and the computer code used in the Field Artillery model are explained in Chapter V. As background material, the STAR model and the Simscript language are described. Each routine and event in the FA program is explained; the variables are described; and a listing of the code is given. Routines and events from the basic STAR model which required modification are presented at the end of the chapter.

Chapter VI presents a typical fire mission from a production run of the model. The term production is used to describe the STAR model with the Field Artillery module added. At selected times in the processing of the mission, a plot was made of the artillery elements involved in the fire mission. These plots are accompanied by the detailed computer printout for that phase of the fire mission.

Chapter VII enumerates possible extensions of the battalion level FA model. Recommendations for enhancements in the areas of target acquisition, command and control, ammunition, and damage assessment are given.

Since the STAR model will become a brigade model, Chapter VIII presents the methodology for extension and modification of the FA model to fit brigade requirements. Additions include counterfires between artillery units, movement of artillery, higher resolution of command and control functions, and additional target acquisition capability.

Appendix A describes the input required to run the model with the current version of STAR. The data are described, the arrays enumerated, and the format discussed. A user should be able to design appropriate input, based on the presentation in this appendix.

The current version of the model is built to portray the Central-European battle in 1986. Assumptions about equipment, tactics, and organizations were based on the experience of the author and, in general, were validated by the United States Training and Doctrine Command. This model is not a full presentation of actual combat: modeling simplifications have been made in several areas. In the opinion of the author, none of these simplifications are critical to the model.

II. THE FIELD ARTILLERY SYSTEM

Success on today's battlefield requires a well trained combined arms team. One of the members of this team is the Field Artillery, whose mission is to suppress, neutralize, or destroy the enemy using cannon, rocket, and missile fires and to integrate all supporting fires into combined arms operations [Ref. 3]. To support this mission, there are three major categories of artillery fires: (1) indirect fires in support of the maneuver forces, (2) counterfires against enemy indirect fire systems, and (3) interdiction fires on the enemy second echelon. The counterfire, interdiction, and fire support coordination functions are not addressed in this version of the model.

Support of maneuver forces requires the artillery to fire where ground elements are fighting the Central Battle. In the European scenario, the enemy in the Central Battle may be characterized in artillery terms as composed of many moving, armored targets. Target location is very difficult for the forward observer (FO) of the fire support team (FIST). The FO must estimate the target's location, speed, direction, and the time until the artillery rounds impact. Based on these assessments, the FO estimates where the target will be when the rounds land. When the rounds do actually detonate, they must be very close to the target to achieve any kind of damage. Due to round-to-round delivery errors, artillery is most appropriate for attack of personnel, thinly armored, and other "soft" targets [Ref. 4]. To the knowledge of this author, the total contribution of the artillery to the Central Battle has not been accurately measured in quantitative terms, particularly in the area of suppression. It is relatively easy

to determine if a target is destroyed but the effect of artillery on the human combat process is more difficult to quantify. For example, even if an artillery round misses an enemy rifleman by 50 meters, the shock, noise, and dust that result may degrade the accuracy of his fire.

The author assumes that there will be different levels of damage by artillery against hard targets. In addition to destroying vehicles, artillery can disrupt or slow a column of enemy vehicles, cause them to disperse, or degrade their vision. The other members of the combined arms team are then in a better position to destroy the targets in that column with direct fire.

The most effective method of accomplishing the Field Artillery mission is through the coordinated employment of all elements of the Field Artillery system. These elements are target acquisition, gunnery, weapons and ammunition, and command and control. Target acquisition is the process of detection, identification, and reporting of the target. Accurate and responsive target acquisition is required if targets are to be effectively attacked. A number of target acquisition devices are available within the Field Artillery system. Human observers, called forward observers (FOs), accompany the maneuver forces on the battlefield. These observers, who comprise a fire support team (FIST), provice both target acquisition and fire support coordination to the company team. (The FIST configuration modeled has one forward observer.) Other means of acquisition available within the Field Artillery system include moving-target radars, indirect fire weapon-locating radars, sound-ranging bases, flash-ranging observation posts, and aerial observers. Outside the Field Artillery, intelligence

agencies collect combat information that can be converted into target information.

The second element, gunnery, is the process of transforming target information into firing data. The fire direction officer (FDO), located in the firing unit fire direction center (FDC), decides how to attack the target. He considers the type and amount of ammunition required, the number of weapons to fire, the method of attack, the accuracy of location, and the time of detection. Upon the completion of this analysis by the FDO, the firing data are computed by a field artillery digital automatic computer (FADAC). Gun crew personnel receive the firing data, load the ammunition, position the weapon, and fire the projectile towards the enemy. Two types of missions are most often fired at the firing unit level. The first is an adjustment mission in which the firing unit begins by firing a single round. When this round lands, the FO spots the location of the impact in relation to the target and sends a correction to the FDC. The FDC computes new firing data and the firing unit fires a new round. When the FO observes the most current round impact close enough to the target that he is confident that he can move the next round onto the target, he indicates that the mission should enter the fire for effect (FFE) phase, in which multiple weapons fire. This type of mission is most appropriate for targets which are moving and targets with a large target location error; however, it is less effective than the second type of mission, which is fire for effect (FFE) with no adjustment. This type mission is appropriate for static targets which have a small target location error and for soft dispersed targets where precision is not required, but in which surprise is essential. The use of FFE

fires and mass fires (in which a number of artillery units fire upon the same target at the same time) maximize the damage to enemy targets.

Field Artillery weapon systems currently available within the armored division are the 155mm howitzer and the 203mm howitzer; the general support rocket system (GSRS) is scheduled to be fielded to the division in 1986. Types of ammunition for cannons include high explosive (HE), dual purpose improved conventional munitions (DPICM), family of scatterable mines (FASCAM), and smoke. As an area fire system, the Field Artillery is limited in its ability to destroy point targets without considerable expenditure of ammunition. Under development, however are 155mm laser-guided projectiles (cannon launched guided projectiles) which will allow the Field Artillery to fire against point targets with a high probability of a first round hit.

The fourth element of the Field Artillery system is command and control. Planning and coordination insure proper allocation of artillery fire, a scarce resource on the battlefield, to support the battle. The allocation of artillery assets among direct support fires, interdiction, and counterfire plays a large part in the successful use of the Field Artillery. Effective command and control ensures that the Field Artillery system provides responsive and accurate fire support.

III. BLUE REPRESENTATION

A. GENERAL

The Blue artillery force was modeled at a high level of resolution because a large data base was available concerning Blue equipment and operational techniques. A number of assumptions and simplifications were made in designing the Blue operations: all were based on the scenario for which the model was initially designed.

1) The specific Field Artillery system modeled was a 'slice' of a direct support FA battalion and its reinforcing artillery. This 'slice' consisted of one 155mm M109A1 battery and one reinforcing 203mm M110A1 battery. The 155mm battery was in direct support (DS) of the Blue battalion. It consisted of two fire direction centers and two four-gun firing units. Each FDC and firing unit worked together as a 'mini-battery'. The 203mm battery consisted of one FDC and one four-gun firing unit. The Blue battalion had three FIST teams, located in the vicinity of the company team commanders that they supported. Target acquisition was the only FIST function modeled. The FIST was configured for an armored division with one FO per company team. Each FO was equipped with both a laser rangefinder and a position locating device. Riding in a special FO vehicle, the FO did not engage any enemy vehicles with direct fire. If the FO and his vehicle were destroyed, the target acquisition function would be transferred to one of the surviving platoon leaders in his company. (In current production runs, the FO was not ever killed because he did not give his position away through direct fire.) This alternate FO would continue to engage the enemy with direct fire and, in addition, would call in artillery; there was no special equipment for the alternate FO.

2) Both HE and DPICM ammunition types were available for the Blue forces. Since the battle modeled was short, all targets were armored, and there were no ammunition constraints, DPICM was used exclusively by the Blue artillery. The commander's guidance was assumed to have specified suppression fires. (It was assumed that four volleys would be sufficient for suppression effects since cannon launched guided projectiles were not available.)

3) Each firing unit fired one mission at a time. Since all targets were moving targets, all missions were adjustment missions.

4) Each FDC was allowed to accept up to two missions to process at any one time. Missions were sent to the reinforcing battery only if both DS FDC's had at least one mission.

5) The firing units did not mass fires. Mass fires are the combined fires of many artillery units against the same target at the same time.

6) The model was designed to allow the user to specify the maximum number of missions that one FO was allowed to process simultaneously. Initial analysis indicated that one way to increase the FO's effect against moving targets was to fire one mission at a time. Because the targets were moving at five to six meters per second, any mission that had to wait became 'stale' very quickly.

7) All missions fired were adjusted by the FO. The FO was allowed to enter fire for effect (FFE) if the radial miss distance of the adjusting round was within a given tolerance (a user input value). A tolerance of 300 meters was chosen for use in production runs because the FO was operating in a defensive situation and had had an opportunity to conduct a terrain analysis. It was felt therefore that the FO could

move the volley's effects onto the target with one final correction.

8) Coordination in the FDC precluded different FOs from firing at the same target. As each mission was processed by an FDC, the target's location was checked against all other current missions. This simulated the fire support coordination that would preclude duplication of targets. The tolerance allowed between the aim-points of fire missions was an input value. If the difference between the aimpoints of two missions was less than this input value, the most recent mission was cancelled. The value used in production runs was 50 meters.

9) A priority system was devised for the targets (groups of vehicles). The priority scheme gave a weight to the target equal to the number of vehicles in the group divided by the distance that these vehicles were from the FO. This elementary weighting scheme was selected because the author felt a small group of targets would seem more dangerous to the FO than a larger group farther away. Since the terms "small", "larger", "close", and "farther" are subjective, the appropriate criteria would be based on military judgement concerning the particular situation. This grouping of vehicles is called a cluster. Figure 1 shows a cluster's priority based on range and number of vehicles in the cluster. The FO selected the group of vehicles with the highest priority to use as his target. The FO was not allowed to fire at a group containing fewer than three vehicles. The number three was chosen for use in production runs because it was felt that this was the minimum number of vehicles that could 'profitably' be engaged.

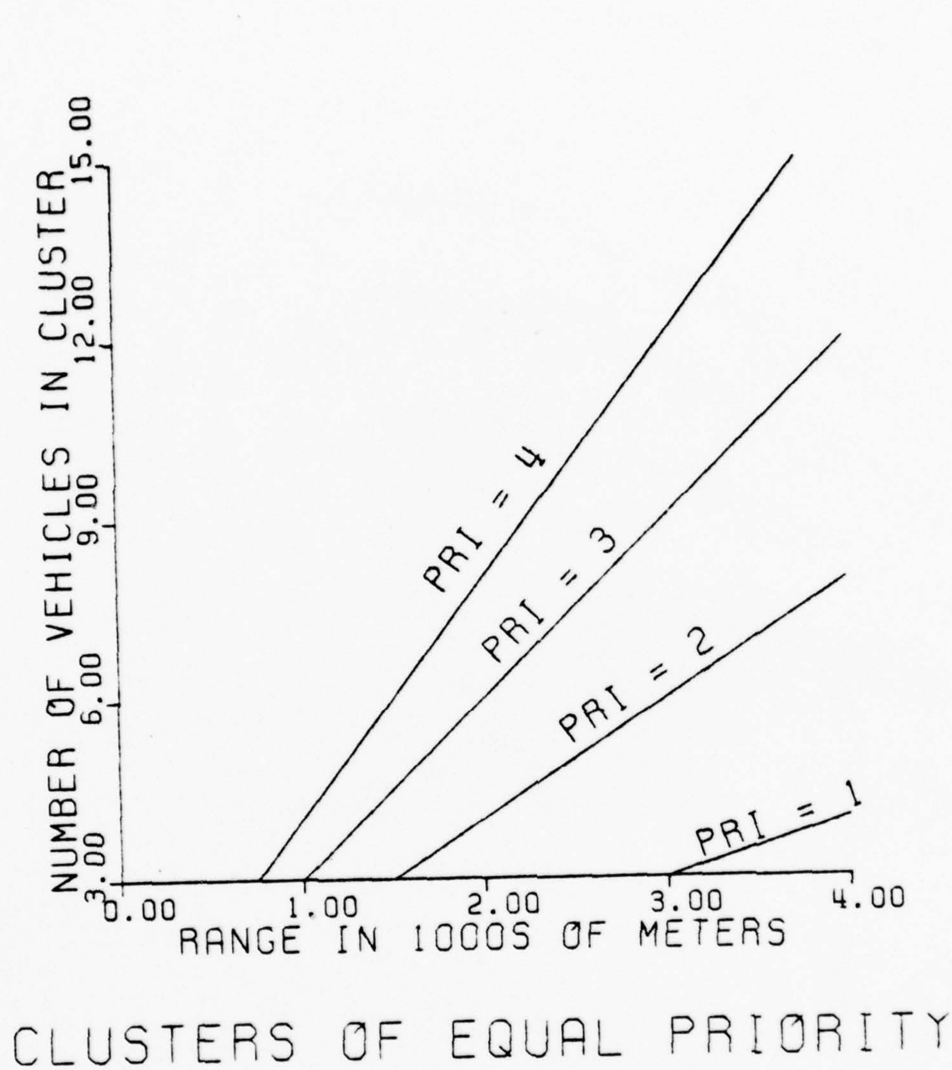


Figure 1. Cluster priority as a function of range and population.

10) TACFIRE equipment was used within all organizations. The digital message device (DMD), a burst transmission device, was used between the FO and the FDC. In the model the FO sent his message to the first DS firing unit FDC, the second DS firing unit FDC, and the reinforcing battery FDC, in that order. The battery computer system (BCS) replaced the FADAC in the FDC.

B. PROCEDURE FOR EACH FORWARD OBSERVER

The STAR model provided a detection model which simulated the procedure by and time at which the observer in a tank or other vehicle acquired enemy vehicles. In the FA model, the FO acquired targets by searching in the same manner as the tank commander, i.e. the FO used routines from STAR to develop a list of targets. The computer simulation checked each vehicle against every other vehicle at given intervals and evaluated the probability of detection (event 'STEP.TIME'). In the Simscript programming language, an event is the computer procedure for accomplishing a number of tasks at a given time. The computer places these tasks in chronological order by 'scheduling' them appropriately. The time of detection was computed as a function of line of sight, range to target, and observer's search pattern. Detection occurred at the appropriate time (event 'DETECT') and the target that was detected was added to the FO's target list (routing 'LIST.UPDATE'). The FO evaluated all detected targets as candidates for a fire mission (event 'CLUSTER. UPDATE'). If the FO was not already processing the maximum number of missions allowed, he aggregated his targets from the target list into clusters. If he had a cluster 'important' enough, he requested a fire mission. The FO prepared a call for fire and attempted to call a fire mission on the battery fire net (a radio frequency dedicated to fires

for the given DS battery) with the battalion FDC (Bn FDC) monitoring, (event 'COMMO.ATTEMPT'). If the DS battery radio frequency was busy, another communications attempt was made five seconds later, (event 'COMMO.ATTEMPT'). Once the communication was received, the Bn FDC TACFIRE would allocate missions to the firing units in the following order: first firing unit and then the second firing unit. If both DS firing units were busy, the call for fire was referred to the reinforcing FDC. If all firing units were busy, a firing unit was selected based on the status of the FDC, the firing unit, and the number of missions waiting to be fired. Once a firing unit was selected, its FDC would start to process the mission (event 'FDC.PROCESSING').

The FDC that received the mission ensured that the firing unit was available to fire the mission, (event 'CHECKING.GUNS.AVAILABILITY'). If the firing unit was busy, then the fire mission was held in a queue until the firing unit was available. If the firing unit was idle, it was scheduled to fire (event 'FIRING.OF.GUNS'). When the adjusting gun was ready to fire, it fired a round that landed at the present time plus the time of flight of the projectile, (event 'ARTY.IMPACT'). The FO was alerted 5 seconds before the round's impact (event 'SPLASH'). The FO determined a new estimated location of the cluster of targets after impact of the adjusting round. When the adjusting round impacted within a given distance of the target, the mission entered the FFE phase. During the FFE phase, the firing unit fired at its maximum rate of fire towards the cluster's estimated location at the predicted time of the impact of the rounds. The FO was, in effect, 'walking' the artillery along the movement vector of the enemy cluster. The FO ended the mission after all volleys had been expended, (event 'END.OF.

MISSION'). The FO then proceeded to detect new vehicles and recluster the previously detected vehicles.

C. CLUSTERING PROCEDURE

The clustering algorithm modeled the actions of the forward observer as he attacked a number of moving armored vehicles. These actions consisted of grouping or 'clustering' vehicles, placing priorities on these clusters, and then estimating the future location of the cluster with the highest priority.

The FO was allowed to detect targets that were in view. These detected targets were stored in his target list until needed. Periodically (a user input value - nominally 30 seconds), the FO clustered his targets. This procedure defined geographical regions ('boxes'), each containing a number of detected vehicles. The model used as a 'box' the rectangular approximation of the effects of one volley fired from a four gun firing unit, (user input - nominally the largest area of effects of the weapons being modeled).

A box was positioned with its center at the coordinates of the first target in the detected list and became a cluster with population of one. The characteristics of a cluster were: 1) X location; 2) Y location; 3) speed, and 4) direction. The initial characteristics of the cluster were those of the first vehicle. The next vehicle detected was then examined. If the vehicle lay within the boundaries of the first cluster's box, the center of the 'box' was moved to the average of the two vehicles' positions, the population increased to two vehicles, and the cluster velocity became the average of the two vehicles' velocity vectors. If the vehicle lay outside the first cluster, it became the center of a new, second cluster. To preclude the box shifting so much

that a previously selected vehicle would fall outside, the box was not allowed to move more than 150 meters in any direction from its original position. Each vehicle in the FO's area of responsibility was placed in an existing cluster or at the center of a new cluster. As more vehicles were added to a cluster, the cluster's original location and velocity were updated appropriately in the manner above. After all vehicles were clustered, each cluster was given a priority.

The priority of a cluster was an indication of its importance as determined by the number of vehicles in the cluster and its closeness to friendly positions. If the population of a cluster was below a given threshold value (initially the value was three), the priority was zero. As a result, the FO would never fire on a cluster of fewer than three vehicles. If the cluster population was greater than or equal to the threshold value, the priority was determined by dividing the number of vehicles in the cluster by the range (in thousands of meters) to the cluster. After each cluster was given a priority, the FO selected the cluster with the highest priority as his target.

Since the cluster was moving, the future location of the cluster at the time of round impact was required for use as an aiming point for the FDC. The time from vehicle identification (clustering) to the impact of the first adjusting round was estimated as the mean of the probability distribution that characterized the time required to get the round to the target. Since the FO was nominally equipped with a laser rangefinding device, the FO lased the cluster centroid, with the assumption made that there would be a single tank at that point. He lased twice at that center tank and 20 seconds later, lased twice again at the same tank in a new location. The average of the first two velocities and the average of the second two velocities were used to

calculate the estimated speed and direction of the cluster. The future location of the vehicle at the time of the round's impact was estimated by extrapolating at the estimated speed in the estimated direction for the estimated time interval. (These calculations will be done by the BCS in the firing unit FDC in conjunction with the FO's lase information.)

D. DAMAGE ASSESSMENT

In the author's opinion, the proper objective of damage assessment is the determination of the effect of artillery rounds placed on or near the target. These effects are detailed as criteria to 'defeat' the specific target under consideration. An example from DARCOM publication 706-101 for tanks is given below [Ref. 4]. This type of defeat criteria was used in the model.

Vulnerability Criteria

K-KILL - A tank was a k-kill (also known as a catastrophic kill) if it was damaged to the extent that it had no further effect on the battle.

F-KILL - A tank was a firepower kill if it was damaged to the extent that it could not fire.

M-KILL - A tank was a mobility kill if it was damaged to the extent that it could not move.

The author made the following definitions. They complement the definitions above but are not supported by available data bases.

NEUTRALIZATION - A tank could not function for some period of time, but returned to the battle later with all capabilities.

SUPPRESSION - A tank had a degradation of some capabilities for some period of time.

The level of damage was used to determine what capabilities to change in the vehicles that were damaged.

Data, based on experiments, are available to provide a basis for damage assessment for the first set of defeat criteria. There are no generally agreed upon definitions for the second set of criteria. Therefore, the model considered only the criteria of k-kill, m-kill and f-kill.

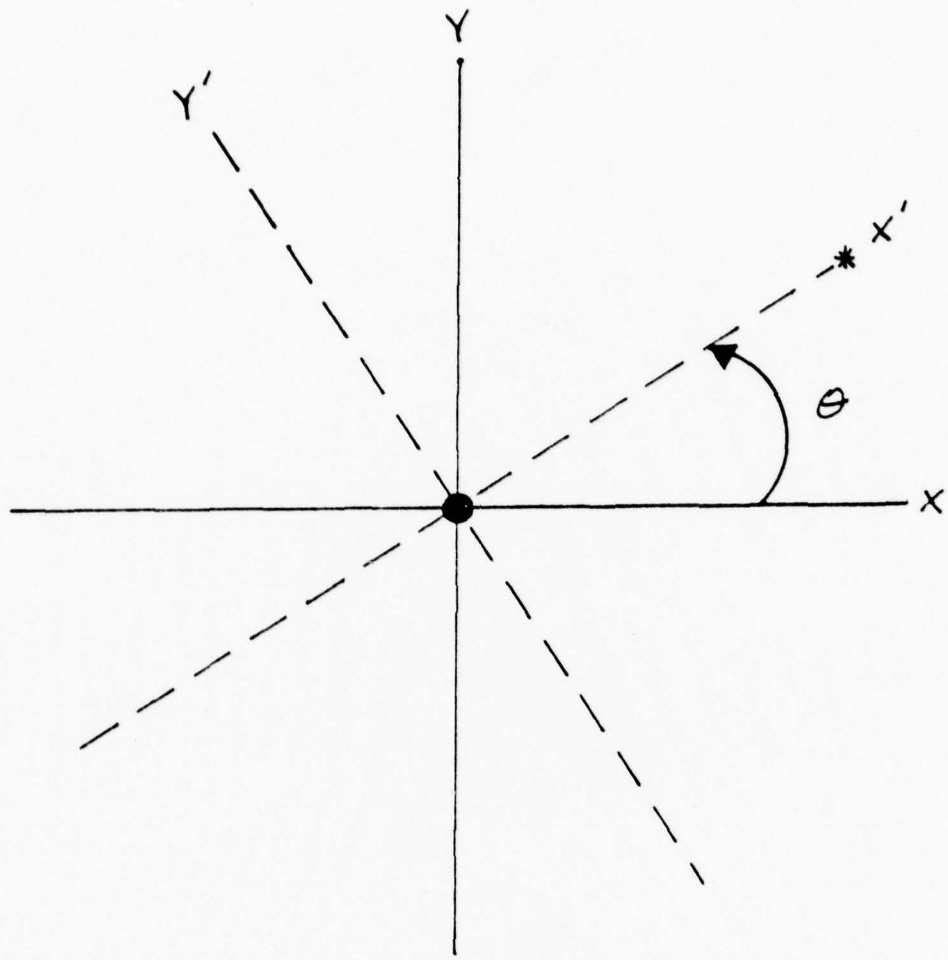
For a given projectile's impact, there is a characteristic pattern of fragmentation. The distribution of HE fragments is spread outward from the sides of the projectile, while the effects of DPICM are circular. In the model, the probability of damaging a target was dependent upon the target's location relative to the round's imapct point. Damage assessment was modeled using the concept of a 'lethal area': the summation of the area around a projectile's impact point weighted by the probability of being damaged while in a given area. Lethal Area (LA) is expressed in the following formula:

$$LA_d = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} P_d \,(x,y)\; dxdy$$

where P_d (x,y) was the conditional probability that a target element located at the point (x,y) on the ground would receive damage at a certain level, d [Ref. 4]. This model used the concept of 'lethal area' with a number of assumptions governing its use. A step function was used to approximate the damage to hard targets with the damage effects

pattern assumed to have a circular shape. If the vehicle lay within a circle centered at the round impact point (radius specified for a given damage level), the probability of receiving that given damage was one, while if the vehicle lay outside the circle, the probability of receiving that given damage was zero.

The Blue damage assessment routine modeled damage as a function of the distance between a round's impact point and the target's location. During the FFE phase, the damage assessment routine calculated the damage inflicted by artillery rounds that had been fired against vehicles in the vicinity of the given aiming point. The mean point of impact (MPI) of all rounds in the volley was displaced from the aiming point in accordance to a bivariate normal distribution based on accuracy probable errors. Each round's actual impact point was moved from its planned aiming point in accordance with range and deflection errors (based on precision probable error). These dispersion errors were assessed in the gun target coordinate system and translated into the battlefield coordinate system. The relationships between the two coordinate systems is shown in Figure 2. The four impact points constituted the volley's effect pattern. Figure 3 shows the effects of one volley fired at a cluster. Each vehicle that was 'alive' was checked to see if it was in the volley's effects pattern. This check compared the vehicle's location to a rectangular approximation of the volley's effects and thereby reduced the number of vehicles that had to be checked later with more elaborate calculations. If a vehicle was within this screening rectangle, its distance from each round's impact point was then calculated. If the vehicle was within a given radius (a user input - nominally 50 meters) of a round's impact point, the



——— Battlefield Coordinate System (x,y)

– – – – – Gun Target Coordinate System (x',y')

● Cluster Centroid (Target)

* Center of Firing Unit (Gun)

θ Theta-Angle through which the coordinate system is rotated

Figure 2. Gun target coordinate system.

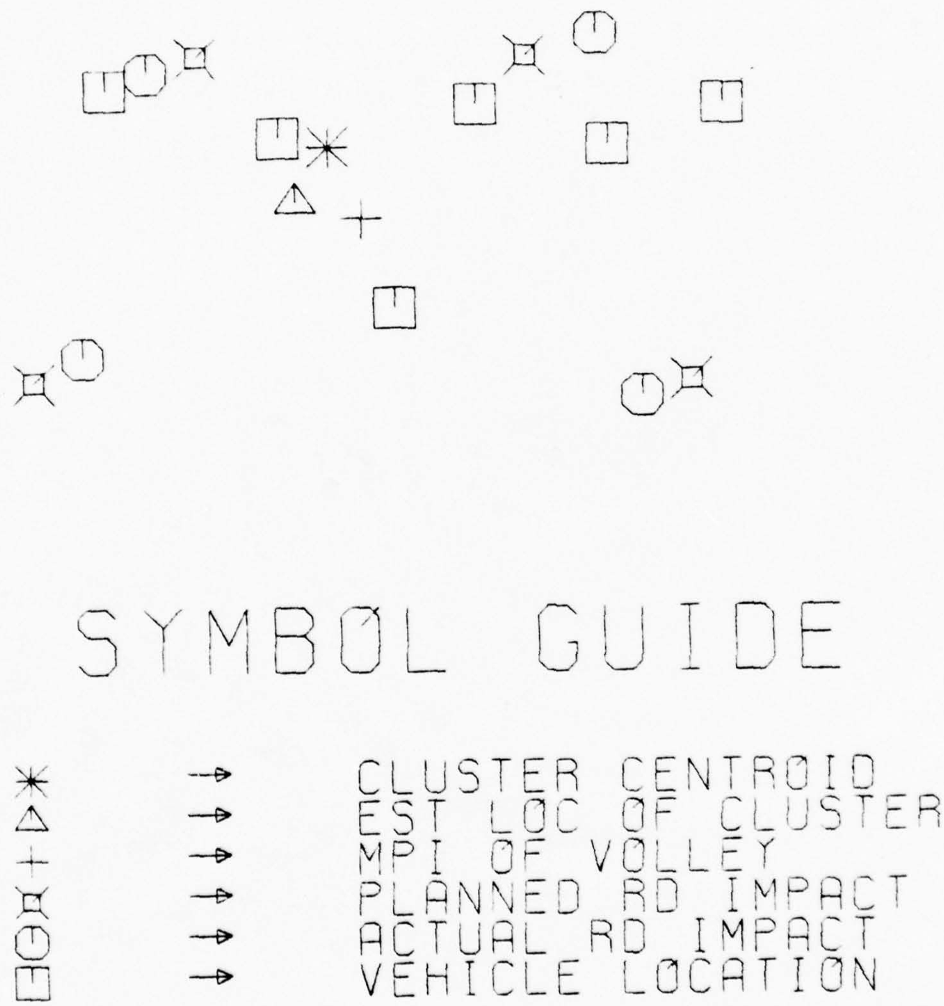


Figure 3. Volley's effects pattern.

distance from the vehicle to the round's impact point was then compared to the appropriate 'lethal radius'. This lethal radius was computed using the appropriate lethal areas and the assumption that the lethal pattern was circular. The k-kill lethal radius was checked first. If the round did not fall within this radius, then the m-kill radius and f-kill radius were both checked. Attributes of the target were altered to reflect any resultant change in capabilities. This procedure was repeated for each round.

IV. RED REPRESENTATION

A. GENERAL

The Red units were more difficult to model than the Blue units due to a lack of relevant information. The Red artillery was modeled at a lower level of resolution. Forward observers were not simulated and no communication or subprocesses were modeled (fire direction center, firing battery, or command and control).

The Red artillery 'attributed slice' in support of the Red regiment consisted of ten 152mm self-propelled howitzer batteries and one BM21 multiple rocket launcher (MRL) battery. By assumption, there were always three of the ten batteries of 152mm artillery moving and seven firing. Lead elements of the Red regiment were assumed to have reported the general locations of the Blue battle positions to the Red regimental headquarters, where the Blue positions were forwarded to the Red artillery headquarters. These Blue battle positions were deemed the regimental objectives and were attacked with continuous supporting fire until the Red maneuver units attacked the front of the Blue battle positions. The planned fires were scheduled to be stopped in 100 seconds when the Red force closed within 1500 meters of any Blue vehicle.

The only Red artillery that was played as 'on call' during the course of the battle was the MRL battery. The Red maneuver commander could request support from the MRL battery to attack Blue targets that were hindering the main attack. Due to their reload time, the MRL were used only once during the attack. The first Red tank that was k-killed by an anti-tank missile fired from the Blue infantry heavy company team

position (team B) signaled the Red commander to request an MRL volley on that Blue location.

B. PLANNED FIRES

The Red artillery fired predominantly preplanned fires. The fire plan was generated external to the model, based on the sheaf of the weapon system and the general Blue positions. Figure 4 shows a fire plan for a Blue company team located in defensive positions over a 900 meter x 1200 meter area. The Red battery's aim point was the center of a 300 meter x 100 meter rectangle, which was used to approximate the effects of a volley from a 152mm battery. After one volley into that rectangle, the battery shifted its fires to the next rectangle, following the pattern indicated on the figure. If the plan was completed, the battery began firing the entire plan a second time. A large target area was divided among the batteries available. Since their strategy depended on volume of fire to defeat the Blue forces, the Red batteries fired at their maximum rate of fire for the duration of the battle.

C. DAMAGE ASSESSMENT

Damage assessment for the Red artillery was at the volley level. Throughout the simulation of Red artillery, it was assumed that there was no overlap of the effects of individual rounds and that those vehicles within the impact area were uniformly distributed over the rectangular area. All Blue vehicles were checked to determine which ones lay within the target rectangle.

The probability of damage was calculated using the following formula (the lethal area for the appropriate target/weapon fired/

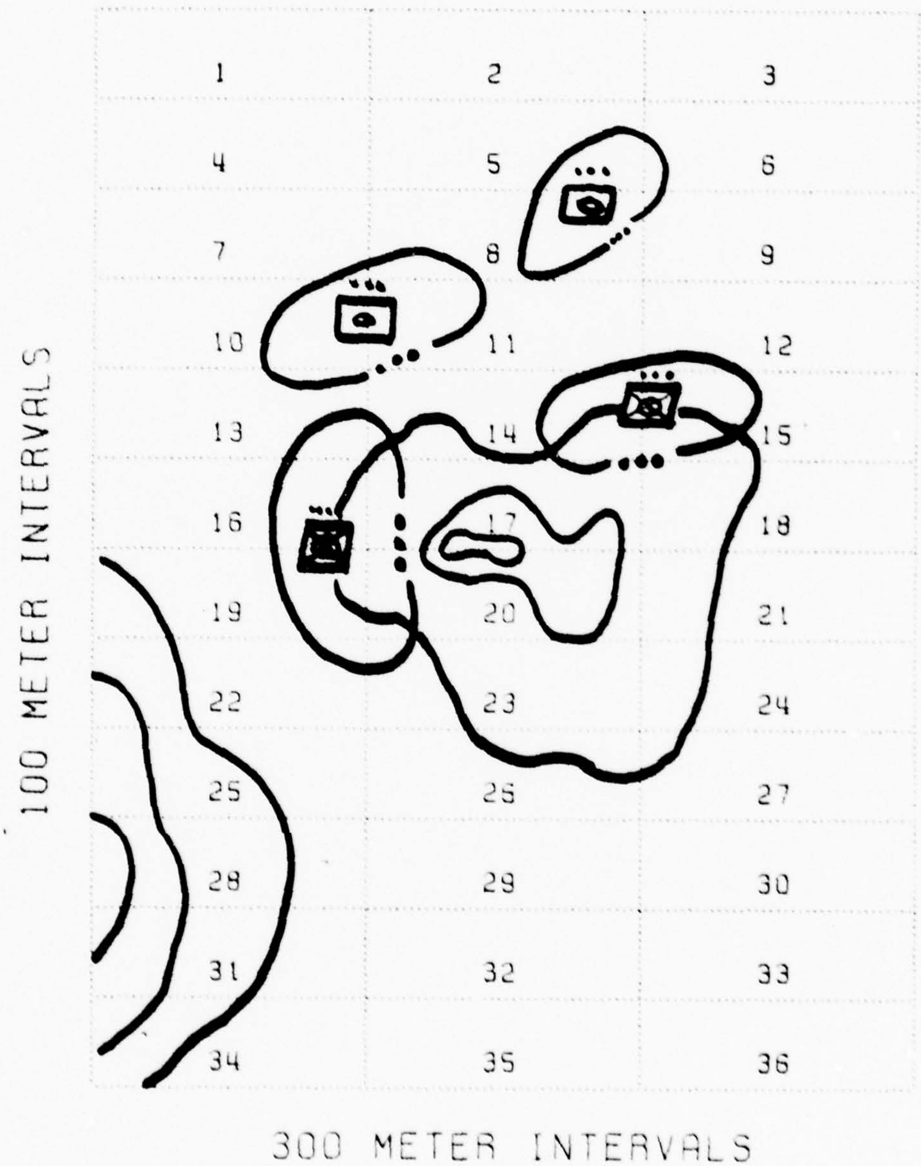


Figure 4. Fire plan for a Red battery.

ammunition was used).

$$\text{Probability of Damage} = \frac{\text{lethal area} \times \text{number of rounds fired}}{\text{area of effects}}$$

The above formula assumes that the area of effects is always much larger than the lethal area times the number of rounds fired. This was always true for all runs performed.

Results were assessed for three levels of damage: k-kill, m-kill, and f-kill. Each target within the target rectangle was evaluated for damage by sending probabilities of damage to the ATRIT routine which assessed the appropriate level of damage to the vehicle. One random number was generated from a uniform distribution (0,1), and was compared to the value of the probability of k-kill that was calculated in routine RED.ARTY.FIRES or event MRL.VOLLEY. If the random number was less than or equal to the probability of a k-kill, a k-kill was assessed. If the vehicle was not k-killed, the incremental m and f damages to this vehicle were accumulated with mobility and firepower damage to this vehicle from other firings. The random number was then compared to the updated m-kill and f-kill thresholds. If the random number was lower in either case, that type kill was assessed against the target and the target's attributes were updated to reflect the results.

V. DETAILED EXPLANATION OF THE FA MODEL

This chapter presents a detailed explanation of the structure of the FA model and its interface with the STAR model. Each routine and event used in the FA model is explained in detail, accompanied by a listing of the code and a definition of the variables.

A. INTERFACE WITH THE STAR MODEL

The STAR model and the FA model were both written in the Simscript language. Simscript was designed for use in discrete event simulations and uses the following concepts: entities, attributes, sets, variables, routines, and events.

An entity is one member of a class of objects, each described by a number of characteristics called attributes. Individual members of the class have specific values assigned to these attributes. There are two types of entities, permanent and temporary. A permanent entity is created at the beginning of the program and exists until the program ends, while a temporary entity is created as required and may be destroyed when no longer needed. When a temporary entity is created, a block of core storage is assigned to it. The beginning address of this core is called the pointer variable of the entity. Permanent entities are assigned an array location at the beginning of the program. As an example of entities and attributes, the entity, MISSION, is an individual fire mission and the attribute, MSN.NAME, is the sequential number of that mission.

Sets are groups of individual entities, ordered in accordance with programmer specifications. In the FA model, all MISSIONS in progress

were placed in the set, HOLDING.MSNS. When the MISSION was completed, it was removed from the set HOLDING.MSNS and placed in the set MSN. QUEUE.

Events are occurrences scheduled to occur at a certain time or after some interval. For details of the Simscript language see reference [6].

The following is a list of the sets, entities, arrays, variables, and routines from the STAR model which are called by the FA model. (The routines and events that required modification are discussed at the end of the chapter.) These must be understood in order to comprehend the code and logic of the FA model [Ref. 1].

System Owned Sets

TANKS - The set to which all direct fire systems belong.

BLUE.ALIVE - The set to which all Blue TANKS initially belong. Blue TANKS are removed when they are killed.

RED.ALIVE - The set to which all Red TANKS initially belong. Red TANKS are removed when they are killed.

Temporary Entities and Their Attributes

TANK - A temporary entity representing a direct fire weapon system.

Attributes of the Tank:

NAME - An integer variable given to a particular TANK for identification.

COLOR - An integer variable with the value of 0 if the color is Red and a value of 1 if the color is Blue.

WPN.TYPE - An integer variable representing a particular type of weapon system.

Integer value	Weapon system name
1	XM1 - 105mm main gun
2	XM1 - 120mm main gun
3	IFV
4	IFV
5	DIVAD
6	DRAGON
7	T72
8	BMP
9	ZSU

ALIVE.DEAD - An integer value with value of 0 if the TANK is alive and a value of 1 if the TANK is dead.

HIT.STATE - An alpha variable, blanks if the vehicle has not been damaged, otherwise an abbreviation of the TANK's status (e.g., DEAD,MFDM,FDAM, etc.).

POINTER - An integer variable which equals the pointer value assigned to the TANK at the time it was created.

X.CURRENT - A real variable representing the TANK's current X coordinate (in meters).

Y.CURRENT - A real variable representing the TANK's current Y coordinate (in meters).

Z.CURRENT - A real variable representing the TANK's current Z coordinate (elevation in meters).

SPD - A real variable representing the TANK's current speed in meters per second.

T.SPD - A real variable representing the simulated battle time at which the TANK's speed was last calculated.

DIR.OF.MVMT - A real variable representing the direction of movement of the TANK in radians, measured with a positive value in the counterclockwise direction from East and negative in the clockwise direction from East.

DEFNUM - An integer variable representing the current defilade conditions of the TANK.

Integer value	Condition
1	full defilade
2	turret defilade
3	firing defilade
4	seeking concealment while stopping to fire
5	unconcealed

OP.RNG - An integer variable representing the maximum range at which the TANK will attempt to engage another TANK. If a vehicle is an FO vehicle, the maximum range of the FO's observation device is stored in the variable.

SEC,PLT,BN,RGT,BDE,DIV - Integer variables representing the identification number of the section, platoon, company, battalion, regiment, brigade, and division to which the TANK is assigned.

MKILL - An integer variable assigned the value of 1 if the TANK had been mobility killed, otherwise equal to zero.

FKILL - An integer variable assigned the value of 1 if the TANK had been firepower killed, otherwise equal to zero.

MFKILL - An integer variable assigned the value of 1 if the TANK had been both mobility and firepower killed, otherwise equal to zero.

KKILL - An integer variable assigned the value of 1 if the TANK had been catastrophically killed, otherwise equal to zero.

COCDR,PLTLDR,SECLDR - Integer variables representing the names of the TANK's company commander, platoon leader, and section leader respectively. Although input as the integer name of the appropriate leader, the attributes are changed by the program to reflect the leader's pointer variable.

Integer Arrays

BBBPOINT - An array containing the pointer variables of all Blue TANKS (direct fire systems).

RRRPOINT - An array containing the pointer variables of all Red TANKS.

LIST - An array containing the pointer variables of the enemy vehicles currently detected by a direct fire weapon system.

Integer Variables

R.NUM.ALIVE - The starting number of Red weapon systems.

B.NUM.ALIVE - The starting number of Blue weapon systems.

BLUE - A variable representing the color of friendly elements.

RED - A variable representing the color of opposing forces.

Routine ATRIT

Arguments:

SH.T - The pointer variable of the weapon which fired. If the source of the potential damage is artillery or mines, SH.T is the pointer variable of the vehicle fired upon.

TGT.T - The pointer of the vehicle which was fired upon.

EMKILL - The probability of an m-kill.

EFKILL - The probability of an f-kill.

KAYKILL - The probability of a k-kill.

WHOCALLED - An integer variable with a value of 0 if a direct fire weapon system caused the possible damage and 1 if a mine or artillery caused the possible damage.

Purpose: To determine the level of damage that a vehicle or weapon system sustains when hit. One random number is drawn from a distribution uniform on the interval 0 to 1. This random number is compared with the probability of a k-kill. If the probability of the k-kill is greater than the value of the random number, a k-kill is assessed. If a k-kill is not assessed, the incremental m and f damages to this vehicle are accumulated with mobility and firepower damage to this vehicle from other firings. The random number is then compared to the current level of m-kill and f-kill damage. If the probability of an m-kill is greater than the random number, an m-kill is assessed. The same procedure is used for the f-kill. If both a mobility and a firepower kill are assessed, an mf-kill is assessed and the vehicle is no longer an active participant in the battle. Upon completion of the assessment, a global variable, DAMAGE.NUM, is assigned one of the following integer values:

Integer value	Type of damage to target
1	no damage
2	mobility damage
3	firepower damage
4	mobility and firepower damage
5	k-kill

<u>Routine DIST</u>

Arguments:

X1 - The current X coordinate of first vehicle.

Y1 - The current Y coordinate of first vehicle.

X2 - The current X coordinate of second vehicle.

Y2 - The current Y coordinate of second vehicle.

Returns: The horizontal distance between two vehicles.

Purpose: To find the distance between two vehicles by using their current battlefield coordinates.

Routine LOC

Argument:

Tank - The pointer variable of the TANK whose location is to be updated.

Purpose: To update the X and Y coordinates of the TANK based on its current speed, direction, and the elapsed time since its last position update. Once the X and Y coordinates are determined, a FORTRAN routine (ELEV) is called which calculates the elevation of the TANK on the parametric terrain model.

Routine TALLY.HIT.STATE

Arguments:

TANK - The pointer value of the TANK that was fired at.

DAMAGE.NUM - The damage number from routine ATRIT or other routines which indicates the type damage sustained by the TANK.

Purpose: Used to accumulate the number of times the TANK was assessed damage in each damage category. When the TANK is assessed as sustaining a k-kill damage, the routine removes the TANK from the BLUE.ALIVE (or RED.ALIVE, as appropriate), PLT.UNIT, and COMP.UNIT sets.

B. FA ROUTINES AND EVENTS

1. PREAMBLE

The PREAMBLE is the first part of any Simscript program. It is used to define global variables and arrays; declare entities, attributes, and sets; and define events and routines. Every variable defined in the PREAMBLE is defined and available throughout the program. Variables defined only in routines and events are local to that particular routine or event. The following items were added to the STAR PREAMBLE. A listing of the additional PREAMBLE coding is provided along with a brief explanation.

System Owned Sets

HOLDING.MSNS - The set to which all current fire missions belong.

MSN.QUEUE - The set to which all completed fire missions belong.

Permanent Entities, Attributes, and Sets

FO - A permanent entity representing the forward observer.

Attributes of Each FO:

POINTING.TO - An integer variable which is equal to the pointer value assigned to the FO's TANK.

MY.RADIO - An integer variable representing the radio frequency on which the FO is operating. A value of 1 indicates the direct support battery radio frequency.

STATUS - An integer variable representing the status of the FO with a value of 0 for idle and 1 for busy.

WAIT.TIME - A real variable equal to the simulated battle time when the FO last attempted to communicate on his radio frequency.

AMT.ACTIVE.MSNS - An integer variable equal to the number of missions in which a FO is currently engaged.

TYPE - An integer variable representing the observation device that the FO is using, with a value of 1 for a laser device and 2 for binoculars. (The observer detects with the unaided eye and then uses the observation device to assist in target location.)

STATE4 - An integer variable used to reference the current mission in the array called C.NUMBER.ARRAY.

LAST.CLUSTERED - A real variable representing the simulated battle time at which the FO last grouped the vehicles that he had detected.

AMT.MSNS.FIRED - An integer variable representing the total number of missions completed by the FO.

<u>BATTERY</u> - A permanent entity representing an artillery firing unit. It owns a set, HOWITZER.QUEUE, which holds all fire missions that are waiting to be fired by that firing unit.

Attributes of Each BATTERY:

COLOR1 - An integer variable whose value is 1 if the color is Blue and 0 if the color is Red.

CALIBER - An integer variable representing the type weapon assigned to the BATTERY. This attribute is assigned the following values: 1 - 155mm, 2 - 203mm, 3 - GSRS, 4 - 152mm, 5 - 122mm MRL.

NUM.GUNS - An integer variable representing the number of weapons in the BATTERY.

X.CUR1 - A real variable representing the BATTERY's current X coordinate in the battlefield coordinate system.

Y.CUR1 - A real variable representing the BATTERY's current Y coordinate on the battlefield coordinate system.

MAX.RANGE - An integer variable representing the maximum firing range of the BATTERY, in meters.

NUM.DPICM.LEFT - An integer variable representing the number of DPICM rounds currently in the BATTERY.

NUM.HE.LEFT - An integer variable representing the number of HE rounds currently in the BATTERY.

NO.MSNS.FIRED - An integer variable representing the number of missions that have been fired by the BATTERY.

STATE1 - An integer variable representing the activity state of the BATTERY, with the following values: 0 - idle, 1 - busy.

ST.FIRING - A real variable representing the time that the BATTERY received the mission, either from the FDC or from the HOWITZER.QUEUE.

QUEUE.SIZE - An integer variable representing the current number of MISSIONs waiting to be fired by the BATTERY.

RATE.OF.FIRE - A real variable representing the maximum rate of fire of the BATTERY, in rounds per minute.

WHICH.VOLLEY - An integer variable representing the current volley being fired in the FFE phase of the MISSION.

KOUNT - An integer variable used only by Red batteries to determine whether to fire DPICM or HE ammunition. Values 1-4 indicate HE ammunition, while a value of 5 indicates DPICM ammunition (indicating an 80% HE basic load, 20% DPICM).

FDC - A permanent entity representing a fire direction center.

Attributes of FDC:

STATE2 - An integer variable representing the status of the FDC with the value of 0 for idle and the value of 1 for busy.

COLOR2 - An integer variable indicating to which force the FDC belongs with a value of 0 for Blue and 1 for Red.

NUM.MISSIONS - An integer variable representing the number of fire missions that have been processed by the FDC.

X.CUR2 - A real variable representing the X coordinate of the FDC.

Y.CUR2 - A real variable representing the Y coordinate of the FDC.

START.PROCESS - A real variable representing the battle time that the FDC started to process the current mission.

RADIO - A permanent entity representing a radio frequency.

Attribute of RADIO:

STATE3 - An integer variable representing the status of the radio frequency. The value of 0 indicates idle and 1 indicates busy.

Temporary Entities, Attributes, and Sets

MISSION - A temporary entity representing a fire mission which is created when the FO has selected a cluster (a group of vehicles) to engage.

Attributes of the MISSION:

MSN.NAME - An integer variable used for identification.

NUM.ADJ.ROUNDS - An integer variable representing the number of rounds required in the adjustment phase.

LEVEL.OF.DAMAGE - An integer variable representing the number of vehicles that were assessed as a k-kill during the MISSION.

FO.TGT.RANGE - A real variable representing the initial distance between the FO and the target (cluster centroid), in meters.

LAST.FO.RG - A real variable representing the final distance (at the last round in FFE) between the FO and the target, in meters.

GT.INITIAL.RG - A real variable representing the initial range between the firing unit and the target.

GT.FINAL.RG - A real variable representing the final range (at the last round in FFE) between the firing unit and the target.

AMT.OF.HITS - An integer variable representing the number of rounds that fell within D.RADIUS distance (user input) from a vehicle during the MISSION.

RD.1.ERROR - A real variable representing the radial error of the first adjusting round from the target.

RD.2.ERROR - A real variable representing the radial error of the final adjusting round from the target if two or more are required. Equal to zero if only one adjusting round is required.

BTRY - An integer variable representing the firing unit firing the MISSION.

AMT.IN.CLUSTER - An integer variable representing the number of vehicles in the cluster that was chosen as a target.

ERROR.CODE - An integer variable indicating what type of error occurred.

Integer value	Error
1	FO waited for over 60 seconds for radio frequency - mission cancelled
2	FO's mission was too close to a mission already being fired (an input value) - mission cancelled.

3	mission was out of range of the firing unit - mission cancelled.
4	(not used)
5	all targets killed by direct fire weapons during adjustment - mission cancelled.
6	all targets killed by direct fire weapons during FFE phase - mission continued.
7	three rounds fired in adjustment - mission cancelled.

MSN.TIME - A real variable representing the total duration of the MISSION. Initially is equal to the time the MISSION started.

QUEUE.TIME - A real variable representing the amount of time that a MISSION waited to be fired after it was received by the firing unit.

DEL.1 - A real variable representing the time between the time the FO clustered and the time that the first adjusting round landed.

DEL.2 - A real variable representing the time duration, in the adjusting phase, between the first rounds impact and the impact of the second round.

FIST - An integer variable representing the FO who is firing the MISSION.

FIRE.DIR.CENTER - An integer variable representing the fire direction center which processes the MISSION.

AMMUNITION.TYPE - An integer variable representing the type of ammunition being fired in the MISSION, with the following values: 1 - DPICM, 2 - HE.

NOW.FIRING - An integer variable representing the status of the MISSION, with the following values: 0 - adjustment phase, 1 - FFE phase.

SPEED - A real variable representing the current speed of the target.

SPD.APPARENT - A real variable representing the speed of the target as estimated by the FO.

DIRECTION - A real variable representing the current direction of movement of the target in radians, measured with a positive value in the counterclockwise direction from East and a negative value clockwise from East.

DIR.APPARENT - A real variable representing the current direction of movement of the target as estimated by the FO, in radians (measured as above).

X.CUR4 - A real variable representing the actual X coordinate of the target.

Y.CUR4 - A real variable representing the actual Y coordinate of the target.

PRI.OF.CLUSTER - A real variable representing the priority of the cluster.

X.MPI - A real variable representing the X coordinate of the location of the mean point of impact of the volley (MPI).

Y.MPI - A real variable representing the Y coordinate of the MPI of the volley.

X.FUTURE.LOC - A real variable representing the X coordinate of the estimated future location of the target.

Y.FUTURE.LOC - A real variable representing the Y coordinate of the estimated future location of the target.

NO.CLUSTER - An integer variable representing the row in the array called CLUSTERS that identifies the current target.

VOLLEYS.TO.FIRE - An integer variable representing the number of volleys that are to be fired by the firing unit during the FFE phase of the MISSION.

THETA - A real variable representing the angle in radians of the gun target line, measured with a positive value in the counter-clockwise direction from East and a negative value clockwise from East.

T.POSITION - A real variable not currently used.

LABEL - An integer variable representing the queue status of the MISSION, with the following values: 0 - not in HOWITZER.QUEUE, 1 - waiting in HOWITZER.QUEUE.

Real Arrays

RD.OFFSET - A two dimensional array which contains the impact coordinates of each round in a volley.

TRAVEL.TIME.ARRAY - A two dimensional array containing the average velocity at two thirds the maximum range of the system for each ammunition/weapon combination.

RANGE.BANDS - A two dimensional array containing the breakpoints of the piecewise linear approximations to the impact point dispersion curves.

FA.TIME.DELTAS - A two dimensional array containing the parameters that characterize the time distributions for all artillery tasks.

TGT.ACQ.ERROR - A two dimensional array containing the parameters that characterize the error distributions of the FO's target acquisition devices, different for each device.

LETHAL.RADIUS.ARRAY - A four dimensional array containing the lethal radius for each weapon type/target type/ammunition type/level of damage.

CLUSTERS - A two dimensional array containing the following information on each cluster of vehicles: the name; number of vehicles in the cluster; priority; the X coordinate, the Y coordinate, the speed and the direction of the cluster; and the X and Y coordinates of the first vehicle placed in the cluster.

DISPLACEMENT - A three dimensional array containing the displacements of the individual firing elements from the center of the firing unit for each weapon type (Blue firing units).

Integer Arrays

C.NUMBER.ARRAY - A three dimensional array containing the name of the cluster in which each detected TANK was placed.

RED.PLANNED.FIRES - A three dimensional array containing the locations of the rectangles being used as targets by the Red firing units.

SIGMA.DPICM - A three dimensional array containing the parameters of the normal distributions that characterize the round dispersion about the impact point. (DPICM ammunition for Blue firing units only).

FO.VEHICLE - A one dimensional array containing the names of the vehicles in which the FOs are riding.

Integer Variables

YES - A variable defined to mean 1.

NO - A variable defined to mean 0.

BUSY - A variable defined to mean 1.

IDLE - A variable defined to mean 0.

ADJ.ROUND - A variable defined to mean 0.

VOLLEY - A variable defined to mean 1.

DPICM - A variable defined to mean 1.

RN.STREAM - A variable equal to a chosen random number stream from Simscript.

DEBUG - A variable representing the amount of FA related computer printout required. A 0 indicates normal output, a 1 indicates that each action of the fire mission be printed, and a value of 5 indicates that the additional printout of the array called CLUSTERS is required.

AMT.RED.BATTERYS - A variable equal to the number of Red firing units created.

AMT.BLUE.BATTERYS - A variable equal to the number of Blue firing units created.

MAX.NUMBER.OF.MISSIONS.PER.FO - A variable equal to the maximum number of missions that an FO is allowed to process simultaneously.

AMT.CALIBERS - A variable equal to the number of different types of artillery weapon systems.

LARGEST.NUM.WEAPONS - A variable equal to the maximum number of weapons in any of the Blue firing units.

AMT.AMMO.TYPES - A variable equal to the number of different types of artillery ammunition.

AMT.FFE.VOLLEYS - A variable equal to the number of volleys to be fired in the FFE phase.

AMT.MRL - A variable equal to the number of MRL batteries.

AMT.FA.TIME.DELTAS - A variable equal to the number of different time parameters used in the array called FA.TIME.DELTAS.

NO.RANGE.BANDS - A variable equal to the number of range bands in the array called RANGE.BANDS.

MISS.TOLERANCE - A variable equal to the minimum radial miss distance for the FO to enter the FFE phase.

FWD.OBS.MSN.TOLERANCE - A variable equal to the minimum acceptable distance between two fire missions. If the distance between two fire missions is closer, the later fire mission is cancelled.

BOX.TOLERANCE - A variable equal to the size of the box used to cluster vehicles.

FO.MAX.RANGE - A variable equal to the maximum range at which an FO is allowed to observe a cluster.

FO.MIN.RANGE - A variable representing the minimum range at which an FO is allowed to observe a cluster.

D.RADIUS - A variable equal to the specified radius within which a vehicle is examined for damage.

SALVOS - A variable equal to the number of volleys that are fired by the Red firing units before shifting to another target.

NUM.MSNS.FIRED - A variable not currently used.

TOW.KOUNT - A variable which indicates whether a MRL volley has occurred, with the value of 0 if no MRL volley has been fired and 1 if the MRL volley has been fired or is in the process of being fired.

LCOUNT - A variable used to number missions as they are created.

RED.OFF - A variable used to represent the status of the RED planned fires. The value of 0 indicates that fires are active, while a value of 1 indicates that the Red forces have closed to within 1500 meters of the Blue positions and the Red planned fires are to be terminated in 100 seconds.

Real Variables

RED.1.CONSTANT - A variable representing the conversion factor to be used for calculating effects of the MRL volley when using the array called ARTY.PK.TABLE.

Coding and Brief Explanation

The coding for the PREAMBLE additions is shown below, followed by a brief explanation where meanings are not obvious.

```
THE SYSTEM HAS A C.NUMBER.ARRAY(*/4)
DEFINE C.NUMBER.ARRAY AS AN INTEGER 3-DIMENSIONAL ARRAY
THE SYSTEM HAS A RED.PLANNED.FIRES(*/4)
DEFINE RED.PLANNED.FIRES AS AN INTEGER 3-DIMENSIONAL ARRAY
THE SYSTEM HAS A SIGMA.DPICM(*/2)
DEFINE SIGMA.DPICM AS AN INTEGER 3-DIMENSIONAL ARRAY
THE SYSTEM OWNS A HOLDING.MSNS AND A MSN.QUEUE
PERMANENT ENTITIES


EVERY FO HAS A POINTING.TO,A MY.RADIO,A STATUS,A WAIT.TIME,
      A AMT.ACTIVE.MSNS, A TYPE, A STATE4,
      A LAST.CLUSTERED AND A AMT.MSNS.FIRED
EVERY BATTERY HAS A COLOR1,A CALIBER,A NUM.GUNS,A X.CUR1,
      A Y.CUR1,A MAX.RANGE,A NUM.DPICM.LEFT,A NUM.HE.LEFT,
      A NO.MSNS.FIRED,A STATE1,A ST.FIRING,A QUEUE.SIZE,
      A RATE.OF.FIRE,A WHICH.VOLLEY,A KOUNT AND OWNS A
      HOWITZER.QUEUE
EVERY FDC HAS A STATE2,A COLOR2,A NUM.MISSIONS,AN X.CUR2,
      A Y.CUR2, AND A START.PROCESS
```

```
EVERY RADIO HAS A STATE 3
      TEMPORARY ENTITIES
EVERY MISSION HAS A MSN.NAME,A NUM.ADJ.ROUNDS,
      A LEVEL.OF.DAMAGE,AN FO.TGT.RANGE,A LAST.FO.RG,
      A GT.INITIAL.RG,A GT.FINAL.RG,A AMT.OF.HITS,
      A RD.1.ERROR,A RD.2.ERROR,A BTRY,A AMT.IN.CLUSTER,
      AN ERROR.CODE,A MSN.TIME,A QUEUE.TIME,A DEL.1,
      A DEL.2, A FIST,A FIRE.DIR.CENTER,
      AN AMMUNTION.TYPE,A NOW.FIRING,A SPEED,
      A SPD.APPARENT,A DIRECTION,A DIR.APPARENT,A X.CUR4,
      A Y.CUR4,A PRI.OF.CLUSTER,AN X.MRI, A Y.MPI,
      A X.FUTURE.LOC,A Y.FUTURE.LOC,A NO.CLUSTER,
      A VOLLEYS.TO.FIRE,A THETA,A T.POSITION,A LABEL
      AND MAY BELONG TO A HOWITZER.QUEUE,A MSN.QUEUE AND
      A HOLDING.MSNS
DEFINE POINTING.TO, MY.RADIO, STATUS,
      AMT.ACTIVE.MSNS,TYPE,STATE4, AND AMT.MSNS.FIRED
      AS INTEGER VARIABLES
DEFINE COLOR1,CALIBER,NUM.GUNS,MAX.RANGE,NUM.OPICM.LEFT,
      NUM.HE.LEFT,NO.MSNS.FIRED,STATE1,QUEUE.SIZE,
      WHICH.VOLLEY AND KOUNT AS INTEGER VARIABLES
DEFINE STATE2,COLOR2 AND NUM.MISSIONS AS INTEGER VARIABLES
DEFINE STATE3  AS AN INTEGER VARIABLE
DEFINE MSN.NAME,NUM.ADJ.ROUNDS,LEVEL.OF.DAMAGE,AMT.OF.HITS,
      BTRY.AMT.IN.CLUSTER,ERROR.CODE,FIST,FIRE.DIR.CENTER,
      AMMUNITION.TYPE,NOW.FIRING,NO.CLUSTER,VOLLEYS.TO.FIRE,
      AND LABEL AS INTEGER VARIABLES
```

```
DEFINE DPICM,VOLLEY,YES,BUSY,ADJ.ROUND,NO AND IDLE AS
     INTEGER VARIABLES
DEFINE DPICM TO MEAN 1
DEFINE VOLLEY TO MEAN 1
DEFINE YES TO MEAN 1
DEFINE BUSY TO MEAN 1
DEFINE ADJ.ROUND TO MEAN 0
DEFINE NO TO MEAN 0
DEFINE IDLE TO MEAN 0
DEFINE RN.STREAM,DEBUG,AMT.RED.BATTERYS,AMT.BLUE.BATTERYS,
     MAX.NUMBER.OF.MISSIONS.PER.FO,AMT.CALIBERS,
     LARGEST.NUM.WPNS,
     AMT.AMMO.TYPES,AMT.FFE.VOLLEYS,AMT.MRL,
     AMT.FA.TIME.DELTAS,NO.RANGE.BANDS,MISS.TOLERANCE,
     FWD.OBS.MSN.TOLERANCE,BOX.TOLERANCE,FO.MAX.RANGE,
     FO.MIN.RANGE,D.RADIUS,SALVOS,NUM.MSNS.FIRED,
     TOW.KOUNT,LCOUNT AND RED.OFF AS INTEGER VARIABLES
DEFINE RED.1.CONSTANT AS A REAL VARIABLE
DEFINE FO.VEHICLE AS AN INTEGER 1-DIMENSIONAL ARRAY
DEFINE RD.OFFSET,TRAVEL.TIME.ARRAY,RANGE.BANDS,
     FA.TIME.DELTAS, AND TGT.ACQ.ERROR AS REAL 2-DIMENSIONAL
     ARRAYS
DEFINE LETHAL.RADIUS.ARRAY AND ARTY.PK.TABLE AS REAL
     4-DIMENSIONAL ARRAYS
DEFINE DISPLACEMENT AND CLUSTERS AS REAL 3-DIMENSIONAL ARRAYS
EVENT NOTICES INCLUDE STOP.SIMULATION,ATTRITION,CHECK AND
SHUT.OFF
EVERY FO.NOT.BUSY HAS A FO7
```

```
186 EVERY UPDATE.CLUSTER HAS A SPEC.FO
187 EVERY COMMO.ATTEMPT HAS A FO6, A MSN6 AND A RAD6
188 EVERY OPEN.RADIO.NET HAS A RAD7
189 EVERY BUSY.RADIO.NET HAS A RAD8
190 EVERY FDC.PROCESSING HAS A NEW.1.FO AND A DIF.MSN
191 EVERY CHECKING.GUNS.AVAILABILITY HAS A BTRY1, A FDC1,
192      A FO1, AND A MSN1
193 EVERY GUNS.FIRING HAS A BTRY2,A FDC2, A FO2, AND A MSN2
194 EVERY ARTY.IMPACT HAS A BTRY3,A FDC3,A FO3, AND A MSN3
195 EVERY END.OF.MISSION HAS A BTRY4,A FDC4,A FO4, AND A MSN4
196 EVERY RED.ARTY.FIRES HAS A IT AND A BTRY5
197 EVERY MRL.IMPACT HAS A X.PLACE AND A Y.PLACE
198 EVERY ALT.FO HAS A NAME9
199 DEFINE FA.1.MAIN,FA.2.MAIN AND PREPLANNED AS RELEASABLE ROUTINES
```

Lines 70, 72, 74 declare C.NUMBER.ARRAY,RED.PLANNED.FIRES, and SIGMA.DPICM as packed integer arrays.

Line 76 defines two sets owned by the system, as opposed to those owned by entities.

Line 199 defines FA.1.MAIN, FA.2.MAIN, and PREPLANNED as releasable routines. (Their core storage can be released after use.)

2. <u>MAIN Program</u>

The MAIN program of the STAR model prepares the model for execution. It reserves space for arrays, sets initial values of global variables, and creates the appropriate Blue and Red systems. It schedules those events that start the simulation. In order to comply with system requirements of the IBM computer that was used, the additions to the main for the FA model were designed as routines to be called from the STAR MAIN.

Coding and Brief Explanation

```
50  CALL FA.1.MAIN
83  CALL FA.2.MAIN
84  RELEASE FA.1.MAIN  RELEASE FA.2.MAIN
```

Lines 50 and 83 call the MAIN routines from the FA model.

Line 84 releases the core storage allocated to routines FA.1.MAIN and FA.2.MAIN, after they have been executed.

3. Routine FA.1.MAIN

Description

Routine FA.1.MAIN partially fulfills the function of the MAIN for the FA module: variables and arrays are read into the program, arrays are dimensioned, entities are created, and attributes are assigned values. This routine is called by the MAIN and calls routine PREPLANNED.

Local Variables

I,J,K,L - Integer variables that are used as counters for do loops.

Coding and Brief Explanation

```
ROUTINE FA.1.MAIN
DEFINE I,J,K AND L AS INTEGER VARIABLES
READ RN.STREAM,DEBUG,FO.MIN.RANGE,FO.MAX.RANGE,
      BOX.TOLERANCE,MISS.TOLERANCE,FWD.OBS.MSN.TOLERANCE
READ N.FO,N.FDC,N.RADIO,N.BATTERY,AMT.BLUE.BATTERYS
      AMT.RED.BATTERYS,MAX.NUMBER.OF.MISSIONS.PER.FO,
      AMT.CALIBERS,NO.RANGE.BANDS,AMT.FA.TIME.DELTAS,
      LARGEST.NUM.WPNS,AMT.AMMO.TYPES,AMT.MRL,
      AMT.FFE.VOLLEYS
RESERVE FO.VEHICLES(*) AS N.FO
```

```
RESERVE FA.TIME.DELTAS (*,*) AS AMT.FA.TIME.DELTAS BY 3
RESERVE SIGMA.DPICM(*,*,*) AS NO.RANGE.BANDS BY 6 BY
     AMT.CALIBERS
RESERVE RANGE.BANDS(*,*) AS NO.RANGE.BANDS BY AMT.CALIBERS
RESERVE CLUSTERS(*,*,*) AS 30 BY 8 BY N.FO
RESERVE C.NUMBER.ARRAY(*,*,*) AS R.NUM.ALIVE BY N.FO BY
     MAX.NUMBER.OF.MISSIONS.PER.FO
RESERVE DISPLACEMENT(*,*,*) AS LARGEST.NUM.WPNS BY 2 BY
     AMT.BLUE.BATTERYS
RESERVE TRAVEL.TIME.ARRAY(*,*) AS AMT.CALIBERS BY
     AMT.AMMO.TYPES
RESERVE ARTY.PK.TABLE(*,*,*,*) AS AMT.CALIBERS BY 9 BY
     AMT.AMMO.TYPES BY 3
RESERVE LETHAL.RADIUS.ARRAY(*,*,*,*) AS AMT.CALIBERS BY 9 BY
     AMT.AMMO.TYPES BY 3
RESERVE TGT.ACQ.ERROR(*,*) AS 2 BY 3
READ TGT.ACQ.ERROR,FO.VEHICLE
LET D.RADIUS = 50
LET SALVOS = 1
CREATE EVERY FO
CREATE EVERY FDC
FOR I = 1 TO N.FDC, DO
READ COLOR2(I),X.CUR2(I),Y.CUR2(I)
LOOP
CREATE EVERY RADIO
CREATE EVERY BATTERY
FOR I = 1 TO N.BATTERY, DO
READ NUM.GUNS(I),COLOR1(I),X.CUR1(I),Y.CUR1(I),
```

```
      NUM.DPICM.LEFT(I),CALIBER(I),MAX.RANGE(I),
      RATE.OF.FIRE(I)
LET NUM.HE.LEFT(I) = 0
LET NUM.DPICM.LEFT(I) = 0
LOOP
FOR L = 1 TO 3, DO
FOR K = 1 TO AMT.AMMO.TYPES, DO
FOR I = 1 TO AMT.CALIBERS, DO
FOR J = 1 TO 9, DO
READ ARTY.PK.TABLE(I,J,K,L)
IF ARTY.PK.TABLE(I,J,K,L) = 0
GO TO LOOP1
ELSE
LET LETHAL.RADIUS.ARRAY(I,J,K,L) =
      SQRT.F(ARTY.PK.TABLE(I,J,K,L)/PI.C)
LET ARTY.PK.TABLE(I,J,K.L) = ARTY.PK.TABLE(I,J,K.L)/30000
'LOOP1' LOOP REPEAT REPEAT REPEAT
LET RED.1.CONSTANT = 240/3
READ TRAVEL.TIME.ARRAY
FOR K = 1 TO AMT.BLUE.BATTERYS, DO
FOR I = 1 TO NUM.GUNS(K), DO
FOR J = 1 TO 2, DO
READ DISPLACEMENT(I,J,K)
LOOP REPEAT REPEAT
READ RANGE.BANDS
FOR K = 1 TO AMT.CALIBERS, DO
FOR I = 1 TO NO.RANGE.BANDS, DO
FOR J = 1 TO 6, DO
```

```
67  READ SIGMA.DPICM(I,J,K)
68  LOOP REPEAT REPEAT
69  READ FA.TIME.DELTAS
70  RESERVE RED.PLANNED.FIRES(*,*,*) AS 30 BY 4 BY
71       AMT.RED.BATTERYS - AMT.MRL
72  CALL PREPLANNED(3,40,3,60,1,0)
73  CALL PREPLANNED(2,39,3,84,1,3)
74  CALL PREPLANNED(2,60,3,35,1,5)
75  RELEASE PREPLANNED
76  RETURN END
```

Lines 1-2 define the routine and the local variables.

Lines 3-9 read in values for global variables.

Lines 10-26 dimension some of the arrays defined in the PREAMBLE.

Line 27 reads in the values for the target acquisition error array and the FO vehicle array.

Lines 28-29 define initial values for PREAMBLE defined global variables.

Line 30 creates each entity that represents a forward observer.

Lines 31-34 create each entity that represents a FDC and read in initial attribute values.

Line 35 creates each entity that represents a radio frequency.

Lines 36-43 create each entity that represents a firing unit and read in initial attribute values.

Lines 44-55 read in all lethal areas, convert to both lethal radii and the probability of damage over a 300 meter X 100 meter area.

Line 56 initializes a conversion factor to be used when a value from array ARTY.PK.TABLE is used to determine MRL lethality.

Line 57 reads in the values of the average velocity of different ammunition.

Lines 58-62 read in the displacement for each weapon from the center of the firing unit.

Line 63 reads in the breakpoints of the linear approximation to the round impace dispersion curves.

Lines 64-68 read in the values of the round impact dispersion at each breakpoint.

Line 69 reads in time parameters.

Lines 70-71 reserve array storage for the array called RED.PLANNED. FIRES.

Lines 72-74 call routine PREPLANNED to create the targets for use in RED planned fires.

Line 75 releases the core storage that had been reserved for routine PREPLANNED.

Line 76 returns control to the MAIN.

4. Routine FA.2.MAIN

Description

Routine FA.2.MAIN completes the function of the MAIN for the FA model. Input values are initialized, selected input values are printed, and initial FO clustering times and Red planned fires are initiated. This routine is called by MAIN and schedules events RED.ARTY.FIRES and UPDATE.CLUSTER.

Local Variables

I - An integer variable used as a counter for a do loop.

Coding and Brief Explanation

```
1 ROUTINE FA.2.MAIN
2 DEFINE I AS AN INTEGER VARIABLE
```

```
FOR I = 1 TO N.FO, DO
LET FA(RRRPOINT(FO.VEHICLE(I))) = I
LET POINTING.TO(I) = RRRPOINT(FO.VEHICLE(I))
LET TYPE(I) = 1
LOOP
PRINT 48 LINES WITH RN.STREAM,DEBUG,FO.MIN.RANGE,
      FO.MAX.RANGE,BOX.TOLERANCE,MISS.TOLERANCE,
      FWD.OBS.MSN.TOLERANCE,MAX.NUMBER,OF.MISSIONS.PER.FO,
      AMT.CALIBERS,NO.RANGE.BANDS,AMT.FA.TIME.DELTAS,
      LARGEST.NUM.WPNS,AMT.AMMO.TYPES,AMT.MRL,
      AMT.FFE.VOLLEYS.B.NUM.ALIVE,R.NUM.ALIVE,N.TANKS,N.FO,
      N.FDO,N.RADIO,N.BATTERY,AMT.BLUE.BATTERYS AND
      AMT.RED.BATTERYS THUS
RN STREAM           =  **
DEBUG               =  **
FO MIN RANGE        = ****.
FO MAX RANGE        = ****.
BOX TOLERANCE       = ****.
MISS TOLERANCE      = ****.
FWD OBS TOLERANCE   = ****.
MAX MSN FO          =  **
AMT CALIBERS        =  **
NO RANGE BANDS      =  **
AMT FA TIME DELTAS  =  **
LARGEST NUM WPNS    =  **
AMT AMMO TYPES      =  **
AMT MRL             =  **
AMT FFE VOLLEYS     =  **
```

```
B NUM ALIVE         =  **
R NUM ALIVE         =  **
N.TANKS             =  **
N.FO                =  **
N.FDC               =  **
N.RADIO             =  **
N.BATTERY           =  **
AMT BLUE BATTERYS   =  **
AMT RED BATTERYS    =  **
    16  FOR I = 1 TO N.FO, DO
    17  SCHEDULE AN UPDATE.CLUSTER(I) IN (10*I) UNITS
    18  LOOP
    19  FOR I = 1 TO AMT.RED.BATTERYS-AMT.MRL, DO
    20  SCHEDULE A RED.ARTY.FIRES(1,AMT.BLUE.BATTERYS+I) IN I*10
    21       UNITS
    22  LOOP
    23  RETURN END
```

Lines 1-2 define the routine and the local variable.

Lines 3-7 initialize attributes of the FO and his vehicle.

Lines 8-15 echo print input values.

Lines 16-18 schedule an initial clustering for each FO.

Lines 19-22 schedule an initial planned fire for each Red battery.

Line 23 returns control to the MAIN.

5. Event FO.NOT.BUSY

Description

Event FO.NOT.BUSY simulates the FO becoming idle after finishing a fire mission. It schedules an UPDATE.CLUSTER at

that time. This event is scheduled by event COMMO.ATTEMPT and event END.OF.MISSION.

Local Variables

ID.FO - An integer variable representing the FO.

Coding and Brief Explanation

```
1  UPON FO.NOT.BUSY(ID.FO)
2  DEFINE ID.FO AS AN INTEGER VARIABLE
3  SCHEDULE AN UPDATE.CLUSTER(ID.FO) NOW
4  IF AMT.ACTIVE.MSNS(ID.FO) = 0
5  LET STATUS(ID.FO) = IDLE
6  ALWAYS
7  RETURN END
```

Lines 1-2 define the event and the local variable.

Line 3 schedules the time of the next clustering for the FO.

Lines 4-6 set the FO's status to idle if he does not have any missions.

Line 7 returns control to the system timer.

6. Event UPDATE.CLUSTER

Description

Event UPDATE.CLUSTER simulates the forward observer's actions during the clustering of vehicles. It calls routine DOING.CLUSTERS to perform the clustering algorithm. Both the name and the priority of the most important cluster are returned as arguments. If the priority is strictly positive, routine NEW.MISSION is called to create a fire mission. The cluster's future location is estimated and the FO attempts to pass the fire mission to the FDC. This event is scheduled by event FA.2.MAIN, event FO.NOT.BUSY and itself. It calls routines

ARTY.TIME,NEW.LOCATION,DOING.CLUSTERS and NEW.MISSION. It schedules event COMMO.ATTEMPT.

Local Variables

ID.FO - An integer variable representing the forward observer.

ID.MISSION - An integer variable representing the newly created mission.

NAME.PRIORITY - An integer variable representing the cluster with the highest priority.

M - An integer variable representing the newly created mission.

TOTAL.CLUSTERS - An integer variable equal to the total number of clusters that the FO created.

TIME.1 - A real variable representing the interval after which the event UPDATE.CLUSTER is rescheduled.

TIME.2 - A real variable representing the time interval after which a communications attempt will be made.

ESTIMATE.OF.TIME - A real variable representing the estimate of the time between the FO's call for fire and the impact of the adjusting round.

PRI.VALUE - A real variable representing the value of the priority of the most important cluster.

Coding and Brief Explanation

```
UPON UPDATE.CLUSTERING(ID.FO)
DEFINE ID.FO,ID.MISSION,NAME.PRIORITY AND M AS INTEGER
     VARIABLES
DEFINE TOTAL.CLUSTERS AS AN INTEGER VARIABLE
DEFINE TIME.1,TIME.2,ESTIMATE.OF.TIME AND PRI.VALUE AS REAL
```

```
     VARIABLES
CALL ARTY.TIME(1) YIELDING TIME.1
SCHEDULE AN UPDATE.CLUSTER(ID.FO) IN TIME.1 UNITS
IF ALIVE.DEAD(POINTING.TO(ID.FO)) = 1
RETURN ELSE
IF M.BLUE.ALIVE (POINTING.TO(ID.FO)) = 0 RETURN ELSE
IF AMT.ACTIVE.MSNS(ID.FO) = MAX.NUMBER.OF.MISSIONS.PER.FO
RETURN ELSE
IF DEBUG GE 1
SKIP 1 OUTPUT LINE
START NEW LINE
PRINT 1 LINE WITH ID.FO AND TIME.V THUS
FO *                 CLUSTERING          TIME = ****.*
SKIP 1 OUTPUT LINE
ALWAYS
LET STATE4(ID.FO) = STATE4(ID.FO) + 1
IF STATE4(ID.FO) = MAX.NUMBER.OF.MISSIONS.PER.FO + 1
LET STATE4(ID.FO) = 1
ALWAYS
LET LAST.CLUSTERED(ID.FO) = TIME.V
CALL DOING.CLUSTERS(ID.FO) YIELDING NAME.PRIORITY,
     PRI.VALUE AND TOTAL.CLUSTERS
IF PRI.VALUE = 0
LET STATE4(ID.FO) = STATE4(ID.FO) - 1
ALWAYS
IF PRI.VALUE GT 0
LET STATUS(ID.FO) = BUSY
CALL NEW.MISSION(ID.FO,NAME.PRIORITY,PRI.VALUE) YIELDING M
```

```
33 LET ID.MISSION = M
34 LET MSN.TIME(ID.MISSION) = TIME.V
35 LET ESTIMATE.OF.TIME = 100
36 'NEXT'
37 CALL NEW.LOCATION(ID.MISSION,ESTIMATE.OF.TIME,1)
38 LET AMT.ACTIVE.MSNS(ID.FO) =
39      AMT.ACTIVE.MSNS(ID.FO) + 1
40 CALL ARTY.TIME(2) YIELDING TIME.2
41 SCHEDULE A COMMO.ATTEMPT(ID.FO,ID.MISSION,1) IN TIME.2 UNITS
42 LET PRI.VALUE = 0
43 ALWAYS
44 RETURN END
```

Lines 1-6 define the event and the local variables.

Lines 7-8 schedule the event at the appropriate time.

Lines 9-10 check whether the FO is dead. If he is dead, control is returned to the system timer.

Line 11 checks whether the FO is dead. If so, control is returned to the system timer. (A different attribute is checked than was checked above.)

Lines 12-13 check whether the FO has the maximum number of missions that he is allowed. If so, control is returned to the system timer.

Lines 14-19 are a print option.

Lines 20-23 increment the value of STATE4 up to the maximum number of missions allowed the FO.

Line 24 updates the time of clustering.

Lines 25-26 call routine DOING.CLUSTERS.

Lines 27-29 check if the priority is zero. If so, the value of STATE4 is reduced by one.

Lines 30-43 check if the value of PRI.VALUE is strictly positive. If so, it sets up the procedures to start a new mission.

Line 44 returns control to the system timer.

7. Event COMMO.ATTEMPT

Description

Event COMMO.ATTEMPT is scheduled by event UPDATE.CLUSTER and by itself. It simulates a radio transmission on a specific frequency. It calls Routine ARTY.TIME for the appropriate time of transmission. The event checks to see if frequency is idle before transmitting. If the frequency is idle, the transmission is received in a DS firing unit FDC at the end of a specific number of seconds. If the frequency is busy, the message is not transmitted and the frequency is checked every 5 seconds to see if it is free. If the FO waits over 60 seconds, the mission is cancelled andthe attribute ERROR.CODE is given the value of 1. The event COMMO.ATTEMPT schedules event OPEN.RADIO.NET, event FDC.PROCESSING, and itself.

Local Variables

ID.FO - An integer variable representing the FO attempting the communication.

ID.MISSION - An integer variable representing the mission being passed to the FDC.

ID.RADIO - An integer variable representing the radio frequency being used.

TIME - A real variable representing the time required to make the transmission, as returned by ARTY.TIME.

Coding and Brief Explanation

```
1  UPON COMMO.ATTEMPT(ID.FO,ID.MISSION,ID.RADIO)
2  DEFINE ID.FO, ID.RADIO AND ID.MISSION AS INTEGER VARIABLES
3  DEFINE TIME AS A REAL VARIABLE
4  IF WAIT.TIME(ID.FO) EQ 0
5  LET WAIT.TIME(ID.FO) = TIME.V
6  ALWAYS
7  IF(TIME.V-WAIT.TIME(ID.FO)) GT 60
8  LET ERROR.CODE(ID.MISSION) = 1
9  LET WAIT.TIME(ID.FO) = 0
10 SCHEDULE A FO.NOT.BUSY(ID.FO) NOW
11 RETURN ELSE
12 IF STATE3(ID.RADIO) = IDLE
13 LET WAIT.TIME(ID.FO) = 0
14 CALL ARTY.TIME(3) YIELDING TIME
15 SCHEDULE A FDC.PROCESSING(ID.FO,ID.MISSION) IN TIME UNITS
16 LET STATE3(ID.RADIO) = 1
17 SCHEDULE AN OPEN.RADIO.NET(ID.RADIO) IN TIME UNITS
18 RETURN ELSE
19 SCHEDULE A COMMO.ATTEMPT(ID.FO,ID.MISSION,ID.RADIO) IN 5
20       UNITS
21 RETURN END
```

Lines 1-3 define the event and local variables.

Lines 4-6 set the value of attribute WAIT.TIME to the current value of the simulation time if this is the first attempt at transmitting.

Lines 7-11 test whether the FO has waited over 60 seconds. If so, control is returned to the system timer.

Line 12 tests if the radio is idle.

Line 13 sets the attribute WAIT.TIME to 0.

Line 14 calls Routine ARTY.TIME for the time parameter.

Line 15 schedules the event FDC.PROCESSING in time seconds.

Line 16 sets the radio to busy.

Lines 17-18 schedule the event OPEN.RADIO.NET in time seconds and control is returned to the system timer.

Lines 19-21 reschedule another COMMO.ATTEMPT in 5 seconds if the radio is busy and control is returned to the system timer.

8. Event OPEN.RADIO

Description

Event OPEN.RADIO simulates a radio frequency becoming available for use. It is scheduled by event COMMO.ATTEMPT, event GUNS.FIRING, event ARTY.IMPACT, and event END.OF.MISSION.

Local Variable

ID.RADIO - An integer variable representing the radio frequency.

Coding and Brief Explanation

```
1 UPON OPEN.RADIO.NET(ID.RADIO)
2 DEFINE ID.RADIO AS AN INTEGER VARIABLE
3 LET STATE3(ID.RADIO) = 0
4 RETURN END
```

Lines 1-2 define the event and the local variable.

Line 3 sets the attribute STATE3 to free.

Line 4 returns control to the calling event.

9. Event BUSY.RADIO

Description

Event BUSY.RADIO simulates a radio frequency becoming busy. It is scheduled by event GUNS.FIRING and event ARTY.IMPACT.

Local Variable

ID.RADIO - An integer variable representing a radio frequency.

Coding and Brief Explanation

```
1 UPON BUSY.RADIO.NET(ID.RADIO)
2 DEFINE ID.RADIO AS AN INTEGER VARIABLE
3 LET STATE3(ID.RADIO) = 1
4 RETURN END
```

Lines 1-2 define the event and the local variable.

Line 3 sets the attribute STATE3 to busy.

Line 4 returns control to the calling event.

10. Event FDC.PROCESSING

Description

Event FDC.PROCESSING simulates a fire direction center processing a fire mission. It checks the target to see if it is too close to any other targets currently being a fired upon. If not, the event checks the target location to insure that it is within range of the selected firing unit. The event is cancelled if any of the tests are failed. Event FDC.PROCESSING is scheduled by event COMMO. ATTEMPT. Event FDC.PROCESSING schedules event CHECKING.GUNS. AVAILABILITY and event END.MISSION and calls routine ARTY.TIME and DIST.

Local Variables

I - An integer variable used as a counter for do loops.

ID.FO - An integer variable representing the FO.

ID.MISSION - An integer variable representing the mission being fired.

ID.BTRY - An integer variable representing the firing unit selected to fire the mission.

DIFF - A real variable representing the range between any two targets that are checked for proximity.

TIME - A real variable representing the time increment until the mission is finished processing in the FDC.

XX - A real variable representing the difference between the X coordinate of the target and the X coordinate of the firing unit.

YY - A real variable representing the difference between the Y coordinate of the target and the Y coordinate of the firing unit.

RG - A real variable representing the range between the target and the firing unit.

<u>Coding and Brief Explanation</u>

```
UPON FDC.PROCESSING(ID.FO,ID.MISSION)
DEFINE I,J,K,L,M,ID.FO,ID.MISSION,ID.BTRY,VOLLEYS,
     AND TYPE.AMMO AS INTEGER VARIABLES
DEFINE DIFF.TIME,YY,RG AND XX AS REAL VARIABLES
IF DEBUG GE 1
SKIP 1 OUTPUT LINE
START NEW LINE
PRINT 1 LINE WITH ID.FO,MSN.NAME(ID.MISSION) AND TIME.V THUS
FO *  ( ** )        FDC PROCESSING       TIME= ****.
FOR I = 1 TO N.FDC, DO
PRINT 1 LINE WITH I,NUM.MISSIONS(I),STATE1(I) AND
     QUEUE.SIZE(I) THUS
             FDC *  ** MSNS ** BTRY ** QUEUE
LOOP
SKIP 1 OUTPUT LINE
ALWAYS
```

```
LET STATUS(ID.FO) = BUSY
LET AMMUNITION.TYPE(ID.MISSION) = DPICM
LET VOLLEYS.TO.FIRE(ID.MISSION) = AMT.FFE.VOLLEYS
FOR I = 1 TO N.FDC, DO
IF NUM.MISSIONS(I) = 0
THEN IF STATE1(I) = 0
GO TO CORRECT.BTRY
ALWAYS LOOP
FOR I = 1 TO N.FDC, DO
IF NUM.MISSIONS(I) = 1
THEN IF STATE1(I) = 1
GO TO CORRECT.BTRY
ALWAYS LOOP
FOR I = 1 TO N.FDC, DO
IF NUM.MISSIONS(I) = 0
THEN IF STATE1(I) = 1
THEN IF QUEUE.SIZE(I) = 0
GO TO CORRECT.BTRY
ALWAYS LOOP
FOR I = 1 TO N.FDC, DO
IF NUM.MISSIONS(I) = 0
THEN IF STATE1(I) = 1
THEN IF QUEUE.SIZE(I) = 1
GO TO CORRECT.BTRY
ALWAYS LOOP
FOR I = 1 TO N.FDC, DO
IF NUM.MISSIONS(I) = 1
THEN IF STATE1(I) = 1
```

```
THEN IF QUEUE.SIZE(I) = 0
GO TO CORRECT.BTRY
ALWAYS LOOP
FOR I = 1 TO N.FDC, DO
IF NUM.MISSIONS(I) = 2
THEN IS STATE1(I) = 0
GO TO CORRECT.BTRY
ALWAYS LOOP
'CORRECT.BTRY'
CALL ARTY.TIME(4) YIELDING TIME
SCHEDULE A CHECKING.GUNS.AVAILABILITY(I,I,ID.FO,ID.MISSIONS) IN
TIME UNITS
LET ID.BTRY = 1
LET XX = X.CUR4(ID.MISSION) - X.CUR1(ID.BTRY)
LET YY = Y.CUR4(ID.MISSION) - Y.CUR1(ID.BTRY)
LET THETA(ID.MISSION) = ARCTAN.F(YY,XX)
CALL DIST(X.CUR1(ID.BTRY),Y.CUR1(ID.BTRY),X.FUTURE.LOC
(ID.MISSION),Y.FUTURE.LOC(ID.MISSION) YIELDING RG
LET GT.INITIAL.RG(ID.MISSION) = RG
LET GT.FINAL.RG(ID.MISSION) = RG
IF RG GT MAX.RANGE(ID.BTRY)
LET ERROR.CODE(ID.MISSION) = 3
SCHEDULE AN END.OF.MISSION(ID.BTRY,ID.BTRY,ID.FO,ID.MISSION) NOW
RETURN ELSE
IF N.HOLDING.MSNS = 0
FILE ID.MISSION IN HOLDING.MSNS
GO TO NEXT
ELSE
```

```
71 FOR EACH MISSION IN HOLDING.MSNS WITH MISSION NE
72      ID.MISSION, DO
73 CALL DIST(X.CUR4(ID.MISSION),Y.CUR4(ID.MISSION),
74      X.CUR4(MISSION),Y.CUR4(MISSION)) YIELDING DIFF
75 IF DIFF LT FWD.OBS.MSN.TOLERANCE
76 LET MSN.TIME(ID.MISSION) = TIME.V - MSN.TIME(ID.MISSION)
77 LET ERROR.CODE(ID.MISSION) = 2
78 SCHEDULE AN END.OF.MISSION(ID.BTRY,ID.BTRY,ID.FO,ID.MISSION) NOW
79 RETURN ELSE
80 LOOP
81 FILE ID.MISSION IN HOLDING.MSNS
82 'NEXT'
83 LET START.PROCESS(I) = TIME.V
84 LET NUM.MISSIONS(I) = NUM.MISSIONS(I) + 1
85 RETURN END
```

Lines 1-4 define the routine and the local variables.

Lines 5-14 are a print option.

Line 15 sets the FO's status to busy.

Line 16 sets the type ammunition to be fired to DPICM.

Line 17 defines the number of volleys to be fired equal to the input value of AMT.FFE.VOLLEYS.

Lines 18-51 check the status of the FDC, firing unit, and the firing unit's waiting queue to determine which firing unit will fire the mission.

Line 52 determines the time interval between the start of FDC.PROCESSING and the time that the firing unit receives the firing data.

Lines 53-54 schedule a check of the firing units status.

Lines 56-58 update THETA.

AMT FFE VOLLEYS = **

Lines 59-60 determine the range between the firing unit and the estimated target location.

Lines 61-62 assign the value of the range to appropriate attributes of MISSION.

Lines 63-66 check whether the target is beyond the range of the firing unit chosen to fire. If so, the attribute ERROR.CODE of MISSION is given the value of three and control is returned to the system timer.

Lines 67-81 check the distance between the centers of targets. If that distance is less than an input value, the attribute ERROR.CODE of MISSION is given the value of 2 and the mission is cancelled. Otherwise, the current mission is placed in the HOLDING.MISSIONS set until the mission is finished.

Lines 83-84 update values of attributes of BATTERY.

Line 85 returns control to the system timer.

11. Event CHECKING.GUNS.AVAILABILITY

Description

Event CHECKING.GUNS.AVAILABILITY checks whether a firing unit is busy when a fire mission is ready to be fired. If the firing unit is busy, the mission is placed in the set, HOWITZER.QUEUE. Otherwise the mission is fired by the firing unit. Event CHECKING .GUNS.AVAILABILITY is scheduled by event FDC.PROCESSING and schedules event GUNS.FIRING. It calls routine ARTY.TIME.

Local Variables

ID.BTRY - An integer variable representing the firing unit.

ID.FDC - An integer variable representing the fire direction center.

ID.FO - An integer variable representing the forward observer.

ID.MISSION - An integer variable representing the fire mission.

TIME - A real variable equal to the time interval between the time the firing unit receives the firing data and the time the firing unit fires.

<u>Coding and Brief Explanation</u>

```
UPON CHECKING.GUNS.AVAILABILITY(ID.BTRY,ID.FDC,ID.FO,ID.MISSION)
DEFINE ID.BTRY,ID.FDC,ID.FO AND ID.MISSION AS INTEGER VARIABLES
DEFINE TIME AS A REAL VARIABLE
IF DEBUG GE 1
SKIP 1 OUTPUT LINE
PRINT 2 LINES WITH ID.FO,MSN.NAME(ID.MISSION),TIME.V,
     ID.BTRY AND STATE1(ID.BTRY) THUS
FO *  ( ** )         CHECKING GUNS        TIME = ****.
                     BTRY *  GUNS *
SKIP 1 OUTPUT LINE
ALWAYS
IF NUM.MISSIONS(ID.FDC) NE 0
LET NUM.MISSIONS(ID.FDC) = NUM.MISSIONS(ID.FDC) - 1
ALWAYS
LET BTRY(ID.MISSION) = ID.BTRY
LET FIRE.DIR.CENTER(ID.MISSION) = ID.FDC
IF STATE1(ID.BTRY) = 0
LET STATE1(ID.BTRY) = 1
LET ST.FIRING(ID.BTRY) = TIME.V
LET NUM.ADJ.ROUNDS(ID.MISSION) = 1
CALL ATRY.TIME(5) YIELDING TIME
SCHEDULE A GUNS.FIRING(ID.BTRY,ID.FDC,ID.FO,ID.MISSION) IN
TIME UNITS
RETURN ELSE
```

FA.2.MAIN, event FO.NOT.BUSY and itself. It calls routines


```
23  LET LABEL(ID.MISSION) = 1
24  IF DEBUG GE 1
25  START NEW LINE
26  PRINT 1 LINE WITH ID.FO,MSN.NAME(ID.MISSION),
27       MSN.NAME(ID.MISSION) AND ID.BTRY THUS
FO * ( ** )       WAITING IN QUEUE   MSN ** FOR BTRY **
28  SKIP 1 OUTPUT LINE
29  ALWAYS
30  LET QUEUE.TIME(ID.MISSION) = TIME.V
31  LET MSN.TIME(ID.MISSION) = 0
32  LET QUEUE.SIZE(ID.BTRY) = QUEUE.SIZE(ID.BTRY) + 1
33  FILE ID.MISSION IN HOWITZER.QUEUE(ID.BTRY)
34  RETURN END
```

Lines 1-3 define the event and the local variables.

Lines 4-9 are a print option.

Lines 10-12 reduce the number of missions being processed by the FDC by 1 if the number is nonzero.

Lines 13-14 assign appropriate values to attributes of the mission.

Lines 15-22 check whether the firing unit is idle. If so, the firing of the guns is scheduled and appropriate attributes are updated.

Line 23 sets value of attribute LABEL to one, indicating that the mission waited in a queue.

Lines 24-29 are a print option.

Lines 30-33 place the mission in a waiting queue (HOWITZER.QUEUE) and assign appropriate values to attributes of MISSION and BATTERY.

Line 34 returns control to the system timer.

DEFINE TIME.1,TIME.2,ESTIMATE.OF.TIME AND PRI.VALUE AS REAL

12. Event GUNS.FIRING

Description

Event GUNS.FIRING simulates the process of firing a single round or volley. A time of flight is calculated based on the range to the target from the firing unit and the ammunition type. The FO is alerted by radio 5 seconds prior to the round/volley impact. Event GUNS.FIRING is schedule by event CHECKING.GUNS.AVAILABILITY, event ARTY.IMPACT, and event END.OF.MISSION. It schedules event ARTY.IMPACT, event OPEN.RADIO.NET and event BUSY.RADIO.NET. It calls routine DIST.

Local Variables

ID.BTRY - An integer variable representing the firing unit.

ID.FDC - An integer variable representing the FDC.

ID.FO - An integer variable representing the FO.

ID.MISSION - An integer variable representing the mission being fired.

WPN.TYPE - An integer variable representing the weapon system of the firing unit.

TYPE.AMMO - An integer variable representing the type of ammunition fired.

TOF - A real variable equal to the time of flight.

RG - A real variable equal to the range in meters between the firing unit and the target.

Coding and Brief Explanation

```
UPON GUNS.FIRING(ID.BTRY,ID.FDC,ID.FO,ID.MISSION)
DEFINE ID.BTRY,ID.FDC,ID.FO AND ID.MISSION,WPN.TYPE AND
     TYPE.AMMO AS INTEGER VARIABLES
DEFINE TOF AND RG AS REAL VARIABLES
```

```
CALL NEW.MISSION(ID.FO,NAME.PRIORITY,PRI.VALUE) YIELDING M
```



```
IF DEBUG GE 1
SKIP 1 OUTPUT LINE
START NEW LINE
PRINT 2 LINES WITH ID.FO,MSN.NAME(ID.MISSION),TIME.V,
     ID.BTRY, AND WHICH.VOLLEY(ID.BTRY) THUS
FO * ( ** )       FIRING GUNS         TIME = ****,
                  BTRY * VOLLEY **
SKIP 1 OUTPUT LINE
ALWAYS
IF NOW.FIRING(ID.MISSION) = ADJ.ROUND
GO TO NEXT
ELSE
LET WHICH.VOLLEY(ID.BTRY) = WHICH.VOLLEY(ID.BTRY) + 1
CALL DIST(X.CUR1(ID.BTRY),Y.CUR1(ID.BTRY),X.FUTURE.LOC
(ID.MISSION),Y.FUTURE.LOC(ID.MISSION)) YIELDING RG
LET GT.FINAL.RG(ID.MISSION) = RG
'NEXT'
LET WPN.TYPE = CALIBER(ID.BTRY)
LET TYPE.AMMO = AMMUNITION.TYPE(ID.MISSION)
LET TOF = GT.FINAL.RG(ID.MISSION)/TRAVEL.TIME.ARRAY(WPN.TYPE,
     TYPE.AMMO)
SCHEDULE AN ARTY.IMPACT(ID.BTRY,ID.FDC,ID.FO,ID.MISSION) IN
     TOF UNITS
IF TOF LT 10 RETURN ELSE
SCHEDULE A BUSY.RADIO.NET(MY.RADIO(ID.FO)) IN TOF-10 UNITS
SCHEDULE AN OPEN.RADIO.NET(MY.RADIO(ID.FO)) IN TOF-5 UNITS
RETURN END
```

STATE4 is reduced by one.

Lines 1-4 define the event and the local variables.

Lines 5-11 are a print option.

Lines 12-15 check whether the battery is in the adjustment phase. If so, control transfers to line 19. Otherwise, the volley count is increased by one.

Lines 16-18 calculate the range to the target and assign it to an attribute of MISSION.

Line 19 is a continuation lable.

Lines 20-23 determine the time of flight and schedule the round/volley impact.

Lines 24-26 check whether the time of flight of the projectile is less than 10 seconds. Normal artillery observed fire procedure is to send the observer a message alerting him that a round is about to land. If the time of flight is less than 10 seconds, control is returned to the system timer, with no message being sent to the FO. Otherwise, an alerting message is sent to the FO. Since a mission in progress has priority over other traffic in a fire control net, the message would be sent over another transmission if the frequency was busy at the time of the transmission of the alert message.

Line 27 returns control to the system timer.

13. <u>Event ARTY.IMPACT</u>

<u>Description</u>

Event ARTY.IMPACT simulates the actions that occur when a round/volley impacts. The event determines the point at which the mission enters the fire for effect phase: Once in the fire for effect phase, the firing is continued until all required rounds are fired. After each volley in the FFE phase, damage is assessed. Event ARTY.IMPACT is scheduled by event GUNS.FIRING,

event END.OF.MISSION, event OPEN.RADIO.NET, and event BUSY. RADIO.NET. It calls routines ERROR,DIST,ARTY.TIME,NEW.LOCATION and ASSESSMENT.

Local Variables

I - An integer variable used as a do loop counter.

ID.BTRY - An integer variable representing the firing unit.

ID.FDC - An integer variable representing the FDC.

ID.FO - An integer variable representing the FO.

ID.MISSION - An integer variable representing the mission being fired.

ANS - A real variable equal to the range in meters from the FO to the target.

RG - A real variable equal to the range in meters from the firing unit to the target.

TIME.1 and TIME.4 - Real variables equal to the time required for the FO to make a subsequent adjustment.

TIME.2 and TIME.5 - Real variables equal to the time required for the communication between the FO and the FDC.

TIME.3 and TIME.6 - Real variables equal to the time required by the firing unit to fire an adjusting round.

TIME.7 - A real variable equal to the time duration between the impact of the last volley and the FO's termination of the mission.

XX - A real variable representing the difference between the X coordinates of the firing unit and the target.

YY - A real variable representing the difference between the Y coordinates of the firing unit and the target.

WITHIN.TOLERANCE - A real variable equal to the radial error between the adjusting round's impact point and the location of the target (or 999 if no vehicles in the cluster are alive).

TIME - A real variable equal to the total time duration from a volley's impact to the next volley being fired.

ESTIMATE.OF.TIME - A real variable equal to the estimated time between the last round in adjustment and the impact of the first volley in FFE.

Coding and Brief Explanation

```
UPON ARTY.IMPACT(ID.BTRY,ID.FDC,ID.FO,ID.MISSION)
DEFINE I,ID.BTRY,ID.FDC,ID.FO AND ID.MISSION AS
      INTEGER VARIABLES
DEFINE ANS,RG,TIME.1,TIME.2,TIME.3,TIME.4,TIME.5,TIME.6,XX,YY,
      WITHIN.TOLERANCE,
TIME,ESTIMATE.OF.TIME AND TIME.7 AS REAL VARIABLES
CALL POSITION.UPDATE(ID.MISSION,0)
IF NOW.FIRING(ID.MISSION) GE VOLLEY
IF DEBUG GE 1
SKIP 1 OUTPUT LINE
PRINT 2 LINES WITH ID.FO,MSN.NAME(ID.MISSION),TIME.V,
      ID.BTRY AND VOLLEYS.TO.FIRE(ID.MISSION) - 1 THUS
FO * ( ** )       ARTY IMPACT        TIME = ****.
SKIP 1 OUTPUT LINE
ALWAYS
GO TO FFE
ELSE
IF NUM.ADJ.ROUNDS(ID.MISSION) = 1
LET DEL.1(ID.MISSION) = TIME.V - MSN.TIME(ID.MISSION)
```

```
ALWAYS
IF NUM.ADJ.ROUNDS(ID.MISSION) = 2
LET DEL.2(ID.MISSION) = TIME.V - DEL.2(ID.MISSION)
ALWAYS
SCHEDULE AN OPEN.RADIO.NET(MY.RADIO(ID.FO)) IN 5 UNITS
SCHEDULE A BUSY.RADIO.NET(MY.RADIO(ID.FO)) NOW
CALL ERROR(ID.BTRY,ID.FDC,ID.FO,ID.MISSION) YIELDING WITHIN.
     TOLERANCE
IF ERROR.CODE(ID.MISSION) = 5
RETURN ELSE
IF NUM.ADJ.ROUNDS(ID.MISSION) = 1
LET RD.1.ERROR(ID.MISSION) = WITHIN.TOLERANCE
JUMP AHEAD
ELSE
LET RD.2.ERROR(ID.MISSION) = WITHIN.TOLERANCE
HERE
IF DEBUG GE 1
SKIP 1 OUTPUT LINE
START NEW LINE
PRINT 2 LINES WITH ID.FO,MSN.NAME(ID.MISSION),TIME.V,
     ID.BTRY,NUM.ADJ.ROUNDS(ID.MISSION) AND
     WITHIN.TOLERANCE THUS
FO * ( ** )        ARTY IMPACT         TIME = ****.
SKIP 1 OUTPUT LINE
ALWAYS
LET XX = X.CUR4(ID.MISSION) - X.CUR1(ID.BTRY)
LET YY = Y.CUR4(ID.MISSION) - Y.CUR1(ID.BTRY)
LET THETA(ID.MISSION) = ARCTAN.F(YY,XX)
```

```
CALL DIST(X.CUR1(ID.BTRY),Y.CUR1(ID.BTRY),X.FUTURE.LOC
(ID.MISSION),Y.FUTURE.LOC(ID.MISSION)) YIELDING RG
LET GT.FINAL.RG(ID.MISSION) = RG
LET NUM.DPICM.LEFT(ID.BTRY) = NUM.DPICM.LEFT(ID.BTRY) - 1
IF WITHIN.TOLERANCE LE MISS.TOLERANCE
IF DEBUG GE 1
SKIP 1 OUTPUT LINE
PRINT 1 LINE WITH ID.FO,MSN.NAME(ID.MISSION) AND TIME.V THUS
FO * ( ** )        ENTERING FFE        TIME = ****.
SKIP 1 OUTPUT LINE
ALWAYS
LET NOW.FIRING(ID.MISSION) = VOLLEY
LET WHICH.VOLLEY(ID.BTRY) = 1
FOR I = 1 TO VOLLEYS.TO.FIRE(ID.MISSION), DO
CALL ARTY.TIME(6) YIELDING TIME.1
CALL ARTY.TIME(3) YIELDING TIME.2
CALL ARTY.TIME(7) YIELDING TIME.3
LET TIME = TIME.1 + TIME.2 + TIME.3
SCHEDULE A GUNS.FIRING(ID.BTRY,ID.FDC,ID.FO,ID.MISSION) IN
TIME + (I-1) * 60/RATE.OF.FIRE(ID.BTRY) UNITS
LOOP
'NEXT2'
LET ESTIMATE.OF.TIME = 78
CALL NEW.LOCATION(ID.MISSION,ESTIMATE.OF.TIME,1)
     RETURN
ELSE
IF NUM.ADJ.ROUNDS(ID.MISSION) = 3
LET ERROR.CODE(ID.MISSION) = 7
```

```
SCHEDULE AN END.OF.MISSION(ID.BTRY,ID.FDC,ID.FO,ID.MISSION) NOW
RETURN
ELSE
LET ESIMATE.OF.TIME = 78
CALL NEW.LOCATION(ID.MISSION,ESTIMATE.OF.TIME,1)
LET DEL.2(ID.MISSION) = TIME.V
LET NUM.ADJ.ROUNDS(ID.MISSION) = NUM.ADJ.ROUNDS(ID.MISSION) + 1
CALL ARTY.TIME(6) YIELDING TIME.4
CALL ARTY.TIME(3) YIELDING TIME.5
CALL ARTY.TIME(7) YIELDING TIME.6
LET TIME = TIME.4 + TIME.5 + TIME.6
SCHEDULE A GUNS.FIRING(ID.BTRY,ID.FDC,ID.FO,ID.MISSION) IN
TIME UNITS
RETURN
'FFE'
CALL ASSESSMENT(ID.BTRY,ID.FDC,ID.FO,ID.MISSION)
LET VOLLEYS.TO.FIRE(ID.MISSION) = VOLLEYS.TO.FIRE(ID.MISSION) - 1
IF VOLLEYS.TO.FIRE(ID.MISSION) = 0
CALL ARTY.TIME(8) YIELDING TIME.7
SCHEDULE AN END.OF.MISSION(ID.BTRY,ID.FDC,ID.FO,ID.MISSION) IN
     TIME.7 UNITS
LET NOW.FIRING(ID.MISSION) = 0
CALL DIST(X.CUR4(ID.MISSION),Y.CUR4(ID.MISSION),X.CURRENT(POINTING.
     TO(ID.FO)),
Y.CURRENT(POINTING.TO(ID.FO))) YIELDING ANS
LET LAST.FO.RG(ID.MISSION) = ANS
CALL DIST(X.CUR4(ID.MISSION),Y.CUR4(ID.MISSION),X.CUR1(ID.BTRY),
Y.CUR1(ID.BTRY)) YIELDING RG
```

```
97  LET GT.FINAL.RG(ID.MISSION) = RG
98  RETURN ELSE
99  LET ESTIMATE.OF.TIME = 60/RATE.OF.FIRE(ID.BTRY)
100 CALL NEW.LOCATION(ID.MISSION,ESTIMATE.OF.TIME,2)
101 RETURN END
```

Lines 1-5 define the event and the local variables.

Line 6 updates the position of the cluster centroid.

Lines 7-15 check if the mission is in the FFE phase. If not, control transfers to line 85 after checking a print option.

Lines 16-21 update attributes of MISSION with the appropriate elapsed time.

Lines 22-23 schedule the events that simulate a radio transmission.

Lines 24-26 determine the radial error of the adjusting round. If ERROR.CODE equals 5 (no live vehicles left in the cluster), control is returned to the system timer.

Lines 27-32 update attribute of MISSION with the radial miss distance.

Lines 33-40 are a print option.

Lines 41-43 update the value of THETA.

Lines 44-47 calculate range from the firing unit to the target and update the ammunition count.

Lines 48-68 check if the radial error is less than a user input value. If so, a print option is checked and the mission goes into the FFE phase. Each volley to be fired is scheduled. The future location of the target is estimated and control is returned to the system timer.

Lines 69-73 check whether the adjusting round count is three. If so, the mission is terminated and control is returned to the system timer.

Lines 74-84 update attributes of the MISSION, schedule the firing of the next adjusting round, and return control to the system timer.

Line 85 is a continuation line.

Line 86 calls the damage assessment routine.

Line 87 updates the count of the volleys left to fire.

Lines 88-98 check if the volley is the last required. If so, the mission is terminated, ranges to the target from the FO and the firing unit are updated, and control is returned to the system timer.

Lines 99-100 update the future estimated location of the target.

Line 101 returns control to the system timer.

14. Event END.OF.MISSION

Description

Event END.OF.MISSION terminates a fire mission, updates attributes of entities, and checks if any missions are waiting for the firing unit just released. If so, the event initiates the firing of the waiting mission. Otherwise, the firing unit reverts to idle status. Event END.OF.MISSION is scheduled by routine ERROR, event FDC.PROCESSING, and event ARTY.IMPACT. It schedules event FO.NOT.BUSY, event OPEN.RADIO.NET, and event GUNS.FIRING. It calls routines ARTY.TIME and NEW.LOCATION.

Local Variables

ID.BTRY - An integer variable representing the firing unit.

ID.FDC - An integer variable representing the FDC.

ID.MISSION - An integer variable representing the mission being fired.

ESTIMATE.OF.TIME - A real variable equal to the estimate of time that themission was delayed.

TIME - A real variable equal to the time duration from the time the FDC sent the mission to the firing unit until the time the rounds were fired.

Coding and Brief Explanation

```
UPON END.OF.MISSION(ID.BTRY,ID.FDC,ID.FO,ID.MISSION)
DEFINE ID.BTRY, ID.FDC, ID.FO AND ID.MISSION AS INTEGER VARIABLES
DEFINE ESTIMATE.OF.TIME AND TIME AS REAL VARIABLES
IF DEBUG GE 1
SKIP 1 OUTPUT LINE
START NEW LINE
PRINT 2 LINES WITH ID.FO,MSN.NAME(ID.MISSION),TIME.V AND
      ID.BTRY THUS
FO * ( ** )        END OF MISSION       TIME = ****.
                   BTRY *
SKIP 1 OUTPUT LINE
ALWAYS
LET WHICH.VOLLEY(ID.BTRY) = 0
LET AMT.MSNS.FIRED(ID.FO) = AMT.MSNS.FIRED(ID.FO) + 1
SCHEDULE AN OPEN.RADIO.NET(MY.RADIO(ID.FO)) NOW
REMOVE ID.MISSION FROM HOLDING.MSNS
IF LABEL(ID.MISSION) = 1
REMOVE ID.MISSION FROM HOWITZER.QUEUE(ID.BTRY)
ALWAYS
LET MSN.TIME(ID.MISSION) = TIME.V - MSN.TIME(ID.MISSION)
FILE ID.MISSION IN MSN.QUEUE
LET NO.MSNS.FIRED(ID.BTRY) = NO.MSNS.FRIED(ID.BTRY) + 1
LET AMT.ACTIVE.MSNS(ID.FO) =
AMT.ACTIVE.MSNS(ID.FO) - 1
IF QUEUE.SIZE(ID.BTRY) GT 0
FOR EACH MISSION IN HOWITZER.QUEUE(ID.BTRY) WITH BTRY(MISSION) =
      ID.BTRY,DO
```

```
GO TO OUT
LOOP
'OUT'
IF DEBUG GE 1
SKIP 1 OUTPUT LINE
START NEW LINE
PRINT 2 LINES WITH FIST(MISSION),MSN.NAME(MISSION),
      TIME.V AND ID.BTRY THUS
FO * ( ** )         LEAVING QUEUE        TIME = ****.
                    BTRY *
SKIP 1 OUTPUT LINE
ALWAYS
LET QUEUE.TIME(MISSION) = TIME.V - QUEUE.TIME(MISSION)
LET MSN.TIME(MISSION) = TIME.V
LET QUEUE.SIZE(ID.BTRY) = QUEUE.SIZE(ID.BTRY) - 1
CALL ARTY.TIME(5) YIELDING TIME
SCHEDULE A GUNS.FIRING(ID.BTRY,ID.FDC,FIST(MISSION),MISSION)
IN TIME UNITS
LET ESTIMATE.OF.TIME = QUEUE.TIME(ID.MISSION)
CALL NEW.LOCATION(MISSION,ESTIMATE.OF.TIME,2)
LET ST.FIRING(ID.BTRY) = TIME.V
LET NUM.ADJ.ROUNDS(MISSION) = 1
IF FIST(MISSION) NE ID.FO
THEN
IF AMT.ACTIVE.MSNS(ID.FO) = 0
SCHEDULE A FO.NOT.BUSY(ID.FO) NOW
ALWAYS
      RETURN
```

```
51 ELSE
52 LET STATE1(ID.BTRY) = 0
53 LET ST.FIRING(ID.BTRY) = 0
54 RETURN END
```

Lines 1-3 define the event and the local variables.

Lines 4-10 are a print option.

Lines 11-13 update attributes of the firing unit and FO. The radio net becomes idle.

Line 14 removes the completed mission from the set of current missions.

Lines 15-17 check if the old mission had waited in a queue. If so, it was taken out of HOWITZER.QUEUE.

Line 18 calculates the elapsed time of the mission.

Line 19 places the completed mission in the completed mission set.

Lines 20-22 update attributes of the firing unit and the FO.

Lines 23-27 check if a mission is waiting to be fired by the firing unit just released. If so, a mission is selected from the queue of waiting missions.

Lines 28-34 are a print option.

Lines 35-37 update attributes of the completed mission and the released firing unit.

Lines 38-40 schedule the firing of the first adjusting round in the new mission.

Lines 41-42 update the estimate of the target's future location.

Lines 43-44 update attributes of the firing unit and the new mission.

Lines 45-51 check if the FO of the new mission is different from the old mission's FO. If so, the old mission's FO becomes idle if he has no more missions to fire. Control is returned to the system timer.

Lines 52-53 update the attributes of the firing unit if there are no missions to be fired.

Line 54 returns control to the system timer.

15. Routine DOING.CLUSTERS

Description

Routine DOING.CLUSTERS simulates the clustering operation of the FO. Detected vehicles are grouped and given priorities. The cluster with the highest nonzero priority is used as the target for a new mission. Routine DOING.CLUSTERS is called by event UPDATE.CLUSTERS. It calls routines DIST and LOC.

Local Variables

I,J.L - Integer variables used as do loop counters.

ID.FO - An integer variable representing the FO.

TOTAL.CLUSTERS - An integer variable equal to the total number of clusters that are formed.

SIZE - An integer variable equal to the number of vehicles that the FO has detected.

TANK - An integer variable representing a detected vehicle.

NAME.PRIORITY - An integer variable representing the cluster with the highest priority.

ANGLE - A real variable equal to the direction of the cluster from the FO's position.

DIR - A real variable equal to the direction of movement of the cluster prior to update with a new vehicle, from the FO's position.

N - A real variable equal to the number of vehicles in the cluster prior to its update.

PRI.VALUE - A real variable equal to the priority of the most important cluster.

X - A real variable equal to the X component of the velocity vector of the cluster being updated.

Y - A real variable equal to the Y component of the velocity vector of the cluster being updated.

B - A real variable equal to the range in meters from the FO to the cluster centroid.

S - A real variable equal to the speed in meters per second of the cluster prior to update.

<u>Coding and Brief Explanation</u>

```
ROUTINE DOING.CLUSTERS (ID.FO) YIELDING NAME.PRIORITY,PRI.VALUE AND
     TOTAL.CLUSTERS
DEFINE I,J AND L AS INTEGER VARIABLES
DEFINE ID.FO, TANK AND NAME.PRIORITY AS INTEGER VARIABLES
DEFINE TOTAL.CLUSTERS AS AN INTEGER VARIABLE
DEFINE SIZE AS AN INTEGER VARIABLE
DEFINE ANGLE,DIR,N,PRI.VALUE,X,B,X AND Y AS REAL VARIABLES
FOR I = 1 TO 30, DO
FOR J = 1 TO 8, DO
LET CLUSTERS(I,J,ID.FO) = 0
LOOP
REPEAT
LET LIST(*) = TARGET(NAME(POINTING.TO(ID.FO)),1)
IF LIST(1) = 0  GO TO OUT.OF.LOOP  ELSE
LET SIZE = DIM.F(LIST(*))
FOR L = 1 TO SIZE, DO
FOR TANK = LIST(L)
```

```
IF ALIVE.DEAD(TANK) = 1
GO TO NEXT
ELSE
CALL LOC(TANK)
LET C.NUMBER.ARRAY(NAME(TANK)-B.NUM.ALIVE,ID.FO,STATE4(ID.FO)) = 0
     IF TOTAL.CLUSTERS = 0
LET C.NUMBER.ARRAY(NAME(TANK)-B.NUM.ALIVE,ID.FO,STATE4(ID.FO)) = 1
     LET CLUSTERS(1,1,ID.FO) = 1
     LET TOTAL.CLUSTERS = 1
     LET CLUSTERS(1,3,ID.FO) = X.CURRENT(TANK)
     LET CLUSTERS(1.4,ID.FO) = Y.CURRENT(TANK)
     LET CLUSTERS(1,5,ID.FO) = SPD(TANK)
     LET CLUSTERS(1,6,ID.FO) = DIR.OF.MVMT(TANK)
     LET CLUSTERS(1,7,ID.FO) = X.CURRENT(TANK)
     LET CLUSTERS(1,8,ID.FO) = Y.CURRENT(TANK)
GO TO NEXT
ELSE
     FOR I = 1 TO TOTAL.CLUSTERS, DO
       IF ABS.F(X.CURRENT(TANK)-CLUSTERS(I,7,ID.FO)) LT BOX.
       TOLERANCE
     THEN
       IF ABS.F(Y.CURRENT(TANK)-CLUSTERS(I,8,ID.FO)) LT BOX.
       TOLERANCE
     LET CLUSTERS(I,1,ID.FO) = CLUSTERS(I,1,ID.FO) + 1
     LET CLUSTERS(I,3,ID.FO) = ((CLUSTERS(I,3,ID.FO)*(CLUSTERS
       (I,1,ID.FO)-1))
+ X.CURRENT(TANK))/CLUSTERS(I,1,ID.FO)
     LET CLUSTERS(I,4,ID.FO) = ((CLUSTERS(I,4,ID.FO)*(CLUSTERS
```

```
(I,1,ID.FO)-1)
+ Y.CURRENT(TANK))/CLUSTERS(I,1,ID.FO)
LET C.NUMBER.ARRAY(NAME(TANK)-B.NUM.ALIVE,ID.FO,STATE4(ID.FO)) = 1
LET N = CLUSTERS(I,1,ID.FO)-1
LET S = CLUSTERS(I,5,ID.FO)
LET DIR = CLUSTERS(I,6,ID.FO)
LET X = ((N*S*COS.F(DIR))+(SPD(TANK)*COS.F(DIR.OF.MVMT(TANK))))/
      (N+1)
LET Y =((N*S*SIN.F(DIR))+(SPD(TANK)*SIN.F(DIR.OF.MVMT(TANK))))/
      (N+1)
LET CLUSTERS(I,5,ID.FO) = SQRT.F(X**2 + Y**2)
IF X = 0
THEN IF Y = 0
LET ANGLE = 0
GO TO ANGLE
ELSE
LET ANGLE = ARCTAN.F(Y,X)
'ANGLE'
LET CLUSTERS(I,6,ID.FO) = ANGLE
GO TO NEXT
ELSE
'FIRST'
'LOOP1'
LOOP
IF C.NUMBER.ARRAY(NAME(TANK)-B.NUM.ALIVE,ID.FO,STATE4(ID.FO) = 0
      LET TOTAL.CLUSTERS = TOTAL.CLUSTERS + 1
LET C.NUMBER.ARRAY(NAME(TANK)-B.NUM.ALIVE,ID.FO,STATE4(ID.FO))
      = TOTAL.CLUSTERS
```

AD-A070 334 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 19/6

A TWO-SIDED FIELD ARTILLERY STOCHASTIC SIMULATION. (U)

MAR 79 S G STARNER

UNCLASSIFIED NL

2 OF 2

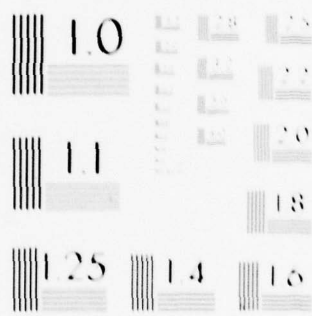
AD A070334

END

DATE FILMED

7-79

DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

```
          LET CLUSTERS(TOTAL.CLUSTERS,1,ID.FO) = 1
          LET CLUSTERS(TOTAL.CLUSTERS,3,ID.FO) = X.CURRENT(TANK)
          LET CLUSTERS(TOTAL.CLUSTERS,4,ID.FO) = Y.CURRENT(TANK)
          LET CLUSTERS(TOTAL.CLUSTERS,5,ID.FO) = SPD(TANK)
          LET CLUSTERS(TOTAL.CLUSTERS,6,ID.FO) = DIR.OF.MVMT(TANK)
     LET CLUSTERS(TOTAL.CLUSTERS,7,ID.FO) = X.CURRENT(TANK)
     LET CLUSTERS(TOTAL.CLUSTERS,8,ID.FO) = Y.CURRENT(TANK)
     ALWAYS
     'NEXT'
     REPEAT
          FOR I = 1 TOTAL.CLUSTERS, DO
          IF CLUSTERS(I,1,ID.FO) GE 3
     CALL DIST(CLUSTERS(I,3,ID.FO),CLUSTERS(I,4,ID.FO),X.CURRENT
          (POINTING.TO(ID.FO)),
     Y.CURRENT(POINTING.TO(ID.FO))) YIELDING B
     IF B LT FO.MIN.RANGE OR B GT FO.MAX.RANGE  GO TO LOOP ALWAYS
     LET CLUSTERS(I,2,ID.FO) = CLUSTERS(I,1,ID.FO)  * 1000/B
          THEN
          IF CLUSTERS(I,2,ID.FO) GT PRI.VALUE
          LET NAME.PRIORITY
          LET PRI.VALUE = CLUSTERS(I,2,ID.FO)
     ALWAYS
     'LOOP'
          LOOP
     IF DEBUG GE 5
     SKIP 1 OUTPUT LINE
     PRINT 2 LINES WITH ID.FO THUS
CLUSTER ARRAY FOR FO# *
```

```
NUM   #      PRI    X      Y      SPD    DIR   XFIR  YFIR
    93  FOR I = 1 TO 20, DO
    94  PRINT 1 LINE WITH I, CLUSTERS(I,1,ID.FO),CLUSTERS(I,2,ID.FO),
    95  CLUSTERS(I,3,ID.FO), CLUSTERS(I,4,ID.FO),CLUSTERS(I,5,ID.FO),
    96  CLUSTERS(I,6,ID.FO), CLUSTERS(I,7,ID.FO), AND CLUSTERS
              (I,8,ID.FO) THUS
 **    ****.  ****.* ****.  ****.  ****.  ****. ****. ****.
    97  LOOP
    98  ALWAYS
    99  'OUT.OF.LOOP'
   100  LET LIST(*) = 0
   101  RETURN END
```

Lines 1-7 define the routine and local variables.

Lines 8-12 initialize values of the array called CLUSTERS to zero.

Lines 13-17 search through the FO's list of detected targets.

Lines 18-20 check if a detected vehicle was previously destroyed. If so, control transfers to line 78.

Line 21 updates location of detected vehicle.

Line 22 initializes to zero the vehicle's representation in the array called C.NUMBER.ARRAY.

Lines 23-34 check whether there are any clusters currently designated. If not, the first cluster is formed with the characteristics of the first detected vehicle. Control is transferred to line 78.

Lines 35-44 screen all clusters to determine if the detected vehicle lies in any of them. If so, the cluster's characteristics are updated.

Lines 45-50 update the velocity vector of the cluster.

Lines 51-57 compute the new direction of the cluster and control transfers to line 78.

Lines 61-61 are continuation labels.

Lines 64-74 check whether the vehicle fell within an existing cluster.

Line 75 is a continuation label.

Lines 77-89 calculate the priority of each cluster.

Lines 90-98 are print options.

Line 99 is a continuation label.

Line 100 releases the FO's list.

Line 101 returns control to the system timer.

16. Routine ASSESSMENT

Description

Routine ASSESSMENT calculates the individual round's impact point. All Red vehicles are then checked to determine which lie within a 300 meter by 300 meter square screening box, with the input point at its center. If a vehicle lies within the screening box, the distance between the vehicle and the impact point is determined. If this difference is less than a given input value, the difference is compared to a lethal radius. If the difference is less than a given lethal radius, damage is assessed at that level. This routine is called by event ARTY.IMPACT. All errors are measured in meters. It calls routines DIST,LOC,NEW. COORDINATE.SYSTEM,PARAMETERS, and TALLY.HIT.STATE.

Local Variables

I and A - Integer variables used as do loop counters.

ID.BTRY - An integer variable representing the firing unit.

ID.FDC - An integer variable representing the FDC.

ID.FO - An integer variable representing the FO.

ID.MISSION - An integer variable representing the mission being fired.

DIFFERENCE - A real variable equal to the radial miss distance between an individual round's impact point and a vehicle's location.

SIG.X - A real variable equal to the standard deviation of range dispersion, measured in the gun target coordinate system.

SIG.Y - A real variable equal to the standard deviation of deflection dispersion, measured in the gun target coordinate system.

X.ERROR - A real variable equal to the range dispersion in meters for the mean point of impact (MPI) of the volley, measured in the gun target coordinate system.

Y.ERROR - A real variable equal to the deflection dispersion in meters for the MPI of the volley, measured in the gun target coordinate system.

X.CHANGE - A real variable equal to the range dispersion of the MPI, measured in the battlefield coordinate system.

Y.CHANGE - A real variable equal to the deflection dispersion of the MPI, measured in the battlefield coordinate system.

X.NORMAL.ERROR - A real variable equal to the range dispersion for an individual round, measured in the gun target coordinate system.

Y.NORMAL.ERROR - A real variable equal to the deflection dispersion for an individual round, measured in the gun target coordinate system.

XNEW - A real variable equal to the X displacement of an individual weapon, measured in the battlefield coordinate system.

YNEW - A real variable equal to the Y displacement of an individual weapon, measured in the battlefield coordinate system.

XDIF - A real variable equal to the range dispersion for an individual round, measured in the battlefield coordinate system.

XDIF - A real variable equal to the deflection dispersion for an individual round, measured in the battlefield coordinate system.

P1 - A real variable equal to the probability of an f-kill.

P2 - A real variable equal to the probability of an m-kill.

P3 - A real variable equal to the probability of a k-kill.

Coding and Brief Explanation

```
ROUTINE ASSESSMENT(ID.BTRY,ID.FDC,ID.FO,ID.MISSION)
DEFINE I,ID.BTRY,ID.FDC,ID.FO,ID.MISSION,
     AND A AS INTEGER VARIABLES
DEFINE DIFFERENCE,SIG.Y,X.ERROR,XDIF,Y.CHANGE,Y.NORMAL.ERROR,
YNEW,SIG.X,X.CHANGE,X.NORMAL.ERROR,XNEW,Y.ERROR,
P1,P2,P3 AND YDIF AS REAL VARIABLES
RESERVE RD.OFFSET(*,*) AS LARGEST,NUM.WPNS BY 2
CALL PARAMETERS(2,AMMUNITION.TYPE(ID.MISSION),CALIBER(ID.BTRY),
GT.FINAL.RG(ID.MISSION)) YIELDING SIG.X AND SIG.Y
LET X.ERROR = NORMAL.F(0,SIG.X,RN.STREAM)
LET Y.ERROR = NORMAL.F(0,SIG.Y,RN.STREAM)
CALL NEW.COORDINATE.SYSTEM(X.ERROR,Y.ERROR,
     THETA(ID.MISSION)) YIELDING X.CHANGE AND Y.CHANGE
LET X.MPI(ID.MISSION) = X.CHANGE + X.FUTURE.LOC(ID.MISSION)
LET Y.MPI(ID.MISSION) = Y.CHANGE + Y.FUTURE.LOC(ID.MISSION)
FOR I = 1 TO NUM.GUNS(ID.BTRY), DO
CALL NEW.COORDINATE.SYSTEM(DISPLACEMENT(I,1,ID.BTRY),
     DISPLACEMENT(I,2,ID.BTRY),THETA(ID.MISSION)) YIELDING
     XNEW AND YNEW
CALL PARAMETERS(1,AMMUNITION.TYPE(ID.MISSION),CALIBER(ID.BTRY),
GT.FINAL.RG(ID.MISSION)) YIELDING SIG.X AND SIG.Y
```

```
LET X.NORMAL.ERROR = NORMAL.F(0,SIG.X,RN.STREAM)
LET Y.NORMAL.ERROR = NORMAL.F(0,SIG.X,RN.STREAM)
CALL NEW.COORDINATE.SYSTEM(X.NORMAL.ERROR,Y.NORMAL.ERROR,
     THETA(ID.MISSION))
YIELDING XDIF AND YDIF
LET RD.OFFSET(I,1) = X.MPI(ID.MISSION) + XNEW + XDIF
LET RD.OFFSET(I,2) = Y.MPI(ID.MISSION) + YNEW + YDIF
LOOP
IF AMMUNITION.TYPE(ID.MISSION) = 1
LET NUM.DPICM.LEFT(ID.BTRY) = NUM.DPICM.LEFT(ID.BTRY) - NUM.GUNS
     (ID.BTRY)
GO TO NEXT
ELSE
LET NUM.HE.LEFT(ID.BTRY) = NUM.HE.LEFT(ID.BTRY) - NUM.GUNS(ID.BTRY)
'NEXT'
FOR EACH TANK IN RED.ALIVE, DO
CALL LOC(TANK)
IF ABS.F(X.CURRENT(TANK) - X.MPI(ID.MISSION)) LT 300
THEN IF ABS.F(Y.CURRENT(TANK) - Y.MPI(ID.MISSION) LT 300
IF DEBUG GE 1
PRINT 1 LINE WITH NAME(TANK) THUS
                ..... TANK # **** WAS FIRED ON  .....
ALWAYS
FOR I = 1 TO NUM.GUNS(ID.BTRY), DO
CALL DIST(X.CURRENT(TANK),Y.CURRENT(TANK),RD.OFFSET(I,1),
     RD.OFFSET(I,2))
YIELDING DIFFERENCE
THEN IF DIFFERENCE LE D.RADIUS
```

```
LET AMT.OF.HITS(ID.MISSION) = AMT.OF.HITS(ID.MISSION) + 1
IF DEBUG GE 1
PRINT 1 LINE WITH I,DIFFERENCE AND NAME(TANK) THUS
        RD# ** LANDED ***. METERS FROM TANK# ***
ALWAYS
FOR A = 1 TO 3, DO
IF DIFFERENCE LE LETHAL.RADIUS.ARRAY(CALIBER(ID.BTRY),
     WPN.TYPE(TANK),AMMUNITION.TYPE(ID.MISSION),A)
GO TO OUT1,OUT2,OUT3 PER A
ELSE
GO TO LOOP2
'OUT1'
LET P3 = 1
GO TO LOOP1
'OUT2'
LET P2 = 1
GO TO LOOP1
'OUT3'
LET P1 = 1
'LOOP1'
CALL ATRIT(TANK,TANK,P1,P2,P3,1)
CALL TALLY.HIT.STATE(TANK,DAMAGE.NUM)
IF DAMAGE.NUM =5
LET ALIVE.DEAD(TANK) = 1
LET FOE(TANK) = C
LET HIT.STATE(TANK) = " FA "
LET LEVEL.OF.DAMAGE(ID.MISSION) = LEVEL.OF.DAMAGE(ID.MISSION) + 1
START NEW LINE
```

```
73 WRITE
74 NCASE, GUNTUBE, AMMUNITION.TYPE(ID.MISSION),NAME(TANK),
75 CALIBER(ID.BTRY),TIME.V.
76      X.CURRENT(TANK),Y.CURRENT(TANK),Z.CURRENT(TANK),SPD(TANK),
77 DEFNUM(TANK),HIT.STATE(TANK)M.D(TANK),F.D(TANK),MKILL(TANK),
78 FKILL(TANK),MFKILL(TANK),
79      KKILL(TANK),K.HIT(TANK),FIRED.AT(TANK)
80 AND NUM.HIT(TANK) AS
81 I 2, I 3,"ARTY", S 8,
82      3 I 4, I 5, S 20, S 7, 3 I 5, D(4,1), I 3, S 11,
83      S 1, A 4, 2 D(5,2), 2 I 2, 5 I 3
84 ALWAYS
85 IF P3 = 1 GO TO OUT4 ELSE
86 'LOOP2'
87 LOOP
88 'OUT4'
89 ALWAYS
90 LOOP
91 REPEAT
92 RELEASE RD.OFFSET(*,*)
93 RETURN END
```

Lines 1-6 define the routine and the local variables.

Line 7 reserve array storage.

Lines 8-9 determine the standard deviations for the MPI dispersion errors.

Lines 10-11 generate the dispersion errors for range and deflection, measured in the gun target coordinate system.

Lines 12-13 change the dispersion errors into the battlefield coordinate system.

Lines 14-15 apply the dispersion errors to the estimated location of the cluster (the planned impact for the volley) to determine the achieved MPI.

Lines 16-28 displace each round from the MPI in accordance with its planned displacement and dispersion errors. The displacement from firing unit center and the dispersion errors are both translated into the battlefield coordinate system.

Lines 29-34 update the ammunition count.

Lines 35-38 screen each Red vehicle. If it les within the screening rectangle, it is tested further.

Lines 39-41 are a print option.

Lines 42-43 determine the miss distance between each round's impact point and the vehicle's location.

Lines 45-46 check if the miss distance is less than or equal to a user input value. If so, an attribute is incremented by one and the miss distance is checked further.

Lines 47-49 are a print option.

Lines 50-66 check the distance determined earlier against three lethal radii. If the distance is less than a given lethal radius, the appropriate probability of damage is given the value of one and the assessment and bookkeeping routines are called.

Lines 67-71 check if the Red vehicle is k-killed. If so, attributes are updated.

Lines 71-83 print out appropriate information on the damage that occurred.

Line 85 checks if the damage was a k-kill. If so, no more checks are made.

Line 92 releases array storage.

Line 93 returns control to the calling event.

17. Routine ERROR

Description

Routine ERROR calculates the distance between an adjusting round and the target. The routine generates the error in range and deflection based on the relative position of the firing unit and the target (gun target coordinate system). A conversion is then made to the battlefield coordinate system. The round's impact point is the estimated location of the target (aiming point given the FDC) with dispersion errors applied. If all vehicles are destroyed before the time of the adjusting round's impact (usually due to direct fire weapons), a value of 5 is given to ERROR.CODE and 999 is returned as an argument. Otherwise, the radial miss distance is returned as an argument. Routine ERROR is called by event ARTY.IMPACT and schedules event END.OF.MISSION. It calls routines DIST, POSITION.UPDATE, and PARAMETERS.

Local Variables

ANS - A real variable equal to the distance between adjusting round and the target.

SIG.X - A real variable equal to the standard deviation parameter for the normal distribution of range dispersion for a single round (precision).

SIG.Y - A real variable equal to the standard deviation parameter for the normal distribution of deflection dispersion (precision).

X.NORMAL.ERROR - A real variable equal to the error in range in terms of the gun target coordinate system.

Y.NORMAL.ERROR - A real variable equal to the error in deflection in terms of the gun target coordinate system.

XDIF - A real variable equal to the error in X coordinates in terms of the battlefield coordinate system.

YDIF - A real variable equal to the error in Y coordinates in terms of the battlefield coordinate system.

X.IMPACT.POINT - A real variable equal to the X coordinate of the adjusting round's impact point.

Y.IMPACT.POINT - A real variable equal to the Y coordinate of the adjusting round's impact point.

Coding and Brief Explanation

```
ROUTINE ERROR(ID.BTRY,ID.FDC,ID.FO,ID.MISSION) YIELDING ANS
DEFINE ID.BTRY,ID.FDC,ID.FO,ID.MISSION AND TANK AS INTEGER
    VARIABLES
DEFINE ANS,SIG.Y,X.IMPACT.POINT,XDIF,Y.NORMAL.ERROR,SIG.X,
    X.NORMAL.ERROR,
Y.IMPACT.POINT AND YDIF AS REAL VARIABLES
CALL PARAMETERS(1,AMMUNITION.TYPE(ID.MISSION),CALIBER(ID.BTRY),
GT.FINAL.RG(ID.MISSION)) YIELDING SIG.X AND SIG.Y
LET X.NORMAL.ERROR = NORMAL.F(0,SIG.X,RN.STREAM)
LET Y.NORMAL.ERROR = NORMAL.F(0,SIG.Y,RN.STREAM)
CALL NEW.COORDINATE.SYSTEM(X.NORMAL.ERROR,Y.NORMAL.ERROR,THETA
    (ID.MISSION))
YIELDING XDIF AND YDIF
LET X.IMPACT.POINT = X.FUTURE.LOC(ID.MISSION) + XDIF
LET Y.IMPACT.POINT = Y.FUTURE.LOC(ID.MISSION) + YDIF
```

```
13   CALL POSITION.UPDATE(ID.MISSION,0)
14   IF X.CUR4(ID.MISSION) = 0
15   THEN IF Y.CUR4(ID.MISSION) = 0
16   LET ERROR.CODE(ID.MISSION) = 5
17   SCHEDULE AN END.OF.MISSION(ID.BTRY,ID.FDC,ID.FO,ID.MISSION) NOW
18   LET ANS = 999
19   GO TO OUT
20   ELSE
21   CALL DIST(X.CUR4(ID.MISSION),Y.CUR4(ID.MISSION),X.IMPACT.POINT,
22   Y.IMPACT.POINT) YIELDING ANS
23   'OUT'
24   RETURN END
```

Lines 1-4 define the routine and the local variables.

Lines 5-6 call the routine PARAMETERS, returning the appropriate standard deviation of the normal distribution describing a round's precision dispersion pattern.

Lines 7-8 generate normal errors in the X and Y directions.

Lines 9-10 transform the normal errors in the gun target coordinate system to errors in the battlefield reference system.

Lines 11-12 apply the errors to the aiming point coordinates.

Line 13 updates the actual location of the cluster centroid.

Lines 14-20 check if the cluster is empty. If so, a value of five is given to the attribute ERROR.CODE of MISSION, an argument of 999 is returned, and control returned to the calling routine.

Lines 21-22 calculate the radial miss distance.

Line 24 returns control to the calling routine.

18. Event RED.ARTY.FIRES

Description

Event RED.ARTY.FIRES assesses the damage to Blue vehicles that lie within the effects pattern of Red planned fires. The probabilities of damage are calculated and routine ATRIT is called to assess damage. Event RED.ARTY.FIRES is scheduled by FA.2.MAIN and itself. It schedules the event ALT.FO and calls routines LOC,ATRIT, and TALLY.HIT.STATE.

Local Variables

ITERATION - An integer variable representing the target number.

I - An integer variable used as a do loop counter.

ID.RED.BTRY - An integer variable representing the firing unit.

TYPE.AMMO - An integer variable representing the type of ammunition fired.

P1 - A real variable equal to the probability of an f-kill.

P2 - A real variable equal to the probability of an m-kill.

P3 - A real variable equal to the probability of a k-kill.

RED.2.CONSTANT - A real variable equal to a conversion factor, used in assessing the probability of damage to a Dragon gunner.

Coding and Brief Explanation

```
UPON RED.ARTY.FIRES(ITERATION,ID.RED.BTRY)
DEFINE ITERATION,I,ID.RED.BTRY AND TYPE.AMMO AS INTEGER
      VARIABLES
DEFINE P1,P2,P3,X AND RED.2.CONSTANT AS REAL VARIABLES
```

```
FOR I = 1 TO SALVOS, DO
IF KOUNT(ID.RED.BTRY) = 5
LET KOUNT(ID.RED.BTRY) = 0
LET TYPE.AMMO = 1
LET NUM.DPICM.LEFT(ID.RED.BTRY) = NUM.DPICM.LEFT(ID.RED.BTRY)-
NUM.GUNS(ID.RED.BTRY)
GO TO NEXT
ELSE
LET TYPE.AMMO = 2
LET NUM.HE.LEFT(ID.RED.BTRY) = NUM.HE.LEFT(ID.RED.BTRY)-
NUM.GUNS(ID.RED.BTRY)
'NEXT'
LET KOUNT(ID.RED.BTRY) = KOUNT(ID.RED.BTRY) + 1
FOR EACH TANK IN BLUE.ALIVE, DO
CALL LOC(TANK)
IF X.CURRENT(TANK) GE RED.PLANNED.FIRES(ITERATION,1,ID.RED.BTRY-3)
       *100
THEN IF X.CURRENT(TANK) LT RED.PLANNED.FIRES(ITERATION,3,ID.RED.
       BTRY-3)*100
THEN IF Y.CURRENT(TANK) GT RED.PLANNED.FIRES(ITERATION,4,ID.RED.
       BTRY-3)*100
THEN IF Y.CURRENT(TANK) LE RED.PLANNED.FIRES(ITERATION,2,ID.RED.
       BTRY-3)*100
LET RED.2.CONSTANT = 1
IF WPN.TYPE(TANK) = 6
THEN IF DEFNUM(TANK) = 3
IF TYPE.AMMO = 1
```

```
LET RED.2.CONSTANT = 5
GO TO ALWAYS
ELSE
LET RED.2.CONSTANT = 10
'ALWAYS'
ALWAYS
LET P1 = ARTY.PK.TABLE(CALIBER(ID.RED.BTRY),WPN.TYPE(TANK),
       TYPE.AMMO,3) * NUM.GUNS(ID.RED.BTRY)
       * RED.2.CONSTANT
LET P2 = ARTY.PK.TABLE(CALIBER(ID.RED.BTRY),WPN.TYPE(TANK),
       TYPE.AMMO,2) * NUM.GUNS(ID.RED.BTRY)
       * RED.2.CONSTANT
LET P3 = ARTY.PK.TABLE(CALIBER(ID.RED.BTRY),WPN.TYPE(TANK),
       TYPE.AMMO,1) * NUM.GUNS(ID.RED.BTRY)
       * RED.2.CONSTANT
CALL ATRIT(TANK,TANK,P1,P2,P3,1)
CALL TALLY.HIT.STATE(TANK,DAMAGE,NUM)
IF DAMAGE.NUM = 5
LET ALIVE.DEAD(TANK) = 1
LET HIT.STATE(TANK) = "FA.1"
LET FOE(TANK) = 0
IF FA(TANK) NE 0
SCHEDULE AN ALT.FO(TANK) IN 100 UNITS
ALWAYS
ALWAYS
START NEW LINE
WRITE
       NCASE, GUNTUBE, TYPE.AMMO, NAME(TANK),
```

```
56 CALIBER(ID.RED.BTRY),TIME.V,
57       X.CURRENT(TNAK),Y.CURRENT(TANK),Z.CURRENT(TANK),SPD(TANK),
58 DEFNUM(TANK),
59 FKILL(TANK),MFKILL(TANK),
60       KKILL(TANK),K.HIT(TANK),FIRED.AT(TANK)
61 AND NUM.HIT(TANK) AS
62 I 2, I 3, "ARTY", S 8,
63       3 I 4, I 5, S 20, S 7, 3 I 5, D(4,1), I 3, S 11,
64       S 1, A 4, 2 D(5,2), 2 I 2, 5 I 3
65 ALWAYS
66 LOOP
67 REPEAT
68 IF ITERATION EQ 30
69 LET ITERATION = 0
70 ALWAYS
71 LET ITERATION = ITERATION + 1
72 SCHEDULE A RED.ARTY.FIRES(ITERATION,ID.RED.BTRY) IN
73 (60/RATE.OF.FIRE(ID.RED.BTRY)) * SALVOS UNITS
74 LET NO.MSNS.FIRE(ID.RED.BTRY) = NO.MSNS.FIRED(ID.RED.BTRY) + 1
75 RETURN END
```

Lines 1-4 define the event and the local variables.

Lines 5-17 determine the ammunition type (20% DPICM and 80% HE) and update ammunition status.

Lines 18-23 check if Blue vehicle lies within the volley's effects.

Lines 24-33 set the value for the conversion factor.

Lines 34-44 calculate the probability of the different damage levels and call the damage assessment and bookkeeping routines.

Lines 45-48 check if vehicle is k-killed. If so, attributes are updated.

Lines 49-51 check if the k-killed system was also an FO. If so, a search for a replacement for the FO is scheduled.

Lines 53-64 print the appropriate damage information.

Lines 68-70 check if the Red fire plan is firing at its last target. If so, the plan is repeated.

Lines 71-73 schedule the next Red planned fire.

Line 74 updates the number of targets that have been fired by the Red battery.

Line 75 returns control to the system timer.

19. Routine NEW.COORDINATE.SYSTEM

Description

Routine NEW.COORDINATE.SYSTEM transforms a position from one coordinate system to the corresponding position in a second coordinate system. It is called by routine ASSESSMENT and routine ERROR.

Local Variables

XNEW - A real variable equal to the X coordinate in the new coordinate system.

YNEW - A real variable equal to the Y coordinate in the new coordinate system.

XOLD - A real variable equal to the X coordinate in the old coordinate system.

YOLD - A real variable equal to the Y coordinate in the old coordinate system.

ANGLE - A real variable equal to the angle between the two coordinate systems, measured counterclockwise in radians from

the horizontal axis.

Coding and Brief Explanation

```
1 ROUTINE FOR NEW.COORDINATE.SYSTEM(XOLD,YOLD,ANGLE) YIELDING XNEW
       AND YNEW
2 DEFINE XNEW AND YNEW AS REAL VARIABLES
3 DEFINE XOLD,YOLD AND ANGLE AS REAL VARIABLES
4 LET XNEW = XOLD * COS.F(ANGLE) - YOLD * SIN.F(ANGLE)
5 LET YNEW = YOLD * COS.F(ANGLE) + XOLD * SIN.F(ANGLE)
6 RETURN END
```

Lines 1-3 define the routine and the local variables.

Line 4 calculates the new X coordinate.

Line 5 calculates the new Y coordinate.

Line 6 returns control to the calling routine.

20. Routine NEW.MISSION

Description

Routine NEW.MISSION creates a mission at the time the FO selects a target to attack. The attributes of the new mission are assigned the appropriate values from the designated cluster plus new values. Routine NEW.MISSION is called by event CLUSTER.UPDATE and calls routine DIST.

Local Variables

ID.FO - An integer variable representing the FO who is firing the mission.

NAME.PRIORITY - An integer variable representing the cluster selected as the target.

M - An integer variable representing the newly created mission.

A - A real variable equal to the range between the FO and the cluster centroid.

PRIORITY - A real variable equal to the priority (relative weight) of the cluster selected by the FO.

<u>Coding and Brief Explanation</u>

```
ROUTINE NEW.MISSION(ID.FO,NAME.PRIORITY,PRIORITY) YIELDING M
DEFINE ID.FO,NAME.PRIORITY AND M AS INTEGER VARIABLES
DEFINE A AND PRIORITY AS REAL VARIABLES
CREATE A MISSION
LET AMT.IN.CLUSTER(MISSION) = CLUSTERS(NAME.PRIORITY,1,ID.FO)
      LET NO.CLUSTER(MISSION) = NAME.PRIORITY
      LET PRI.OF.CLUSTER(MISSION) = PRIORITY
      LET X.CUR4(MISSION) = CLUSTERS(NAME.PRIORITY,3,ID.FO)
      LET Y.CUR4(MISSION) = CLUSTERS(NAME.PRIORITY,4,ID.FO)
      LET SPEED(MISSION) = CLUSTERS(NAME.PRIORITY,5,ID.FO)
      LET DIRECTION(MISSION) = CLUSTERS(NAME.PRIORITY,6,ID.FO)
LET LCOUNT = LCOUNT + 1
LET MSN.NAME(MISSION) = LCOUNT
LET FIST(MISSION) = ID.FO
CALL DIST(X.CUR4(MISSION),Y.CUR4(MISSION),X.CURRENT(POINTING.TO
      (ID.FO)),
Y.CURRENT(POINTING.TO(ID.FO))) YIELDING A
LET FO.TGT.RANGE(MISSION) = A
LET M = MISSION
IF DEBUG GE 1
START NEW LINE
PRINT 3 LINES WITH ID.FO,MSN.NAME(MISSION),SPEED(MISSION),X.CUR4
      (MISSION),
```

```
   22  Y.CUR4(MISSION),DIRECTION(MISSION) AND A THUS
FO *         COMPUTING MSN              MSN ***
             SPD **. X ****.         Y ****. DIR **.*
             FO RANGE TO TGT ****.
   23  SKIP 1 OUTPUT LINE
   24  ALWAYS
   25  RETURN END
```

Lines 1-3 define the routine and the local variables.

Line 4 creates a new MISSION.

Lines 5-14 give values to attributes of the new MISSION.

Lines 15-17 determine the distance in meters from the FO to the cluster centroid.

Line 18 assigns the value of the pointer of the MISSION to the argument M.

Lines 19-24 are a print option.

Line 25 returns control to the calling event.

21. Routine PARAMETERS

Description

Routine PARAMETERS calculates the standard deviation for range and deflection from linear approximations of the standard deviation of dispersion about the impact point (stored in array SIGMA.DPICM). Array RANGE.BANDS contains the endpoints (breakpoints) of each of the linear approximations. The routine interpolates the appropriate value. If the range is less than the minimum value in the array, the minimum value is used. If the range is greater than the maximum value in the array, the maximum value is used. Routine PARAMETERS is called by routine ERROR and routine ASSESSMENT.

Local Variables

I - An integer variable representing the value of COUNT.

J - An integer variable representing the type of ERROR required, with a value of 1 for precision error and 2 for error in MPI.

K - An integer variable representing the caliber of the weapon firing the MISSION.

TYPE.AMMO - An integer variable representing the type of ammunition used in the MISSION, with a value of 1 as DPICM and 2 as HE.

COUNT - An integer variable representing the upper value of a pair of endpoints that bracket the range.

RG - A real variable equal to the range from the firing unit to the cluster centroid. The value is converted to range in thousands of meters.

DELTA.1 - A real variable equal to the difference in the endpoints that bracket the range.

DELTA.2 - a real variable equal to the difference between the range and the lower of the two endpoints that bracket the range.

SIG.DF - A real variable equal to the standard deviation for deflection.

SIG.RG - A real variable equal to the standard deviation for range.

FRACTION - A real variable equal to the distance along the linear approximation for which the interpolation was required.

Coding and Brief Explanation

```
ROUTINE PARAMETERS(J,TYPE.AMMO,K,RG) YIELDING SIG.RG AND SIG.DF
DEFINE I,J,K,COUNT AND TYPE.AMMO AS INTEGER VARIABLES
DEFINE DELTA.2,SIG.DF,DELTA.1,FRACTION AND SIG.RG AS REAL VARIABLES
DEFINE RG AS A REAL VARIABLE
LET RG = RG/1000
IF RG LE RANGE.BANDS(1,K)
LET COUNT = 2
LET FRACTION = 0
GO TO NEXT
ELSE
IF RG GE RANGE.BANDS(NO.RANGE.BANDS,K)
LET COUNT = NO.RANGE.BANDS
LET FRACTION = 1
GO TO NEXT
ELSE
FOR COUNT = 2 TO NO.RANGE.BANDS, DO
IF RG LT RANGE.BANDS(COUNT,K)
GO TO NEXT
LOOP
ALWAYS
'NEXT'
LET I = COUNT
LET DELTA.1 = RANGE.BANDS(COUNT,K) - RANGE.BANDS(COUNT-1,K)
LET DELTA.2 = RG - RANGE.BANDS(COUNT-1,K)
LET FRACTION = DELTA.2/DELTA.1
LET SIG.RG = SIGMA.DPICM(I-1,2*J-1,K)+FRACTION * (SIGMA.DPICM
      (I,2*J-1,K)
SIGMA.DPICM(I-1,2*J,K))
```

```
28  LET SIG.DF = SIGMA.DPICM(I-1,2*J,K) + FRACTION * (SIGMA.DPICM
         (I,2*J,K)-
29  SIGMA.DPICM(I-1,2*J,K))
30  RETURN END
```

Lines 1-4 define the routine and the local variables.

Line 5 calculates the range in thousands of meters.

Lines 6-10 check if the range is less than the lowest breakpoint. If it is, control transfers to line 21.

Lines 11-15 check if the range is greater or equal to the largest endpoint. If it is, control is transferred to line 21.

Lines 16-20 set the value of COUNT to the value of the position in the array called RANGE.BANDS of the highest endpoint of the set of endpoints that bracket the range.

Line 21 is a continuation line.

Lines 22-29 access the array called SIGMA.DPICM and interpolate as necessary.

Line 30 returns control to the calling routine.

22. Routine ARTY.TIME

Description

Routine ARTY.TIME generates an elapsed time required to accomplish different tasks based on probability distributions of the time durations. The routine uses the array called FA. TIME.DELTAS, a two dimensional array with the tasks identified through the first subscript and the values of the task in the second subscript. There are three values stored for each task. The first value identifies the type distribution and has the following values: 1-deterministic, 2-uniform distribution, and 3-normal distribution. If the distribution is deterministic,

its value is in the second column. If the distribution is uniform, the start point of the interval is in column two and the end point is in column three. If the distribution is normal, the value of the mean is in column two and the value of the standard deviation is in column three. The routine uses the Simscript functions UNIFORM.F and NORMAL.F to generate the appropriate numbers based on the chosen random number stream. Routine ARTY.TIME is called by event UPDATE.CLUSTER, event COMMO.ATTEMPT, event FDC.PROCESSING, event CHECKING.GUNS.AVAILABILITY, event FIRING.GUNS, event ARTY.IMPACT, and event END.OF.MISSION.

Local Variable

A - An integer variable representing the type of task:

integer value	task
1	Update cluster
2	FO calculations
3	Commo from FO to FDC
4	Processing in FDC
5	Firing unit operations
6	FO subsequent adjusting round
7	Firing unit operations - subsequent rd
8	End of mission

DEL.TIME - A real variable equal to the elapsed time generated from parameters of the array called FA.TIME.DELTAS.

Coding and Brief Explanation

```
1  ROUTINE ARTY.TIME(A) YIELDING DEL.TIME
2  DEFINE DEL.TIME AS A REAL VARIABLE
3  DEFINE A AS AN INTEGER VARIABLE
4  IF FA.TIME.DELTAS(A,1) = 1
5  LET DEL.TIME = FA.TIME.DELTAS(A,2)
6  RETURN ELSE
7  IF FA.TIME.DELTAS(A,1) = 2
8  LET DEL.TIME = UNIFORM.F(FA.TIME.DELTAS(A,2),FA.TIME.DELTAS(A,3),
         RN.STREAM)
9  RETURN ELSE
10 LET DEL.TIME = NORMAL.F (FA.TIME.DELTAS(A,2),FA.TIME.DELTAS(A,3),
         RN.STREAM)
11 LET DEL.TIME = MAX.F(0,DEL.TIME)
12 RETURN END
```

Lines 1-3 define the routine and the local variables.

Lines 4-6 apply if the distribution is deterministic. The time is generated and control returned to the calling event.

Lines 7-9 apply if the distribution is uniform. If it is, the time is generated and control returned to the calling event.

Lines 10-12 generate a time from a normal distribution and return control to the calling event.

23. <u>Routine FA.TGT.ERROR</u>

<u>Description</u>

Routine FA.TGT.ERROR generates a random error for an FO's sighting (estimation of location) of a moving vehicle. It uses array TGT.ACQ.ERROR, a 2 by 3 array. The rows correspond to the value of A, the type of observation device. The first column of the appropriate row identifies the type distribution and has

the following values: 1 - deterministic, 2 - uniform distribution, and 3 - normal distribution. If the distribution is deterministic, its value is in column two. If the distribution is uniform, the start point of the interval is in column two and the endpoint is in column three. If the distribution is normal, the value of the mean is in column two and the value of the standard deviation is in column three. The routine uses the Simscript functions UNIFORM.F and NORMAL.F to generate the appropriate random numbers based on the chosen random number stream. Routine FA.TGT.ERROR is called by routine NEW.LOCATION.

Local Variables

A - An integer variable representing the FO's observation device with a value of 1 for a laser ranging system and 2 for binoculars(device used for location and not detection).

LOC.ERROR - A real variable equal to the target location error in meters.

Coding and Brief Explanation

```
ROUTINE FA.TGT.ERROR(A) YIELDING LOC.ERROR
DEFINE A AS AN INTEGER VARIABLE
DEFINE LOC.ERROR AS A REAL VARIABLE
IF TGT.ACQ.ERROR(A,1) = 1
LET LOC.ERROR = TGT.ACQ.ERROR(A,2)
RETURN ELSE
IF TGT.ACQ.ERROR(A,1) = 2
LET LOC.ERROR = UNIFORM.F(TGT.ACQ.ERROR(A,2),
TGT.ACQ.ERROR(A,3),RN.STREAM)
RETURN ELSE
LET LOC.ERROR = NORMAL.F(TGT.ACQ.ERROR(A,2),
```

```
12 TGT.ACQ.ERROR(A,3),RN.STREAM)
13 RETURN END
```

Lines 1-3 define the routine and the local variables.

Lines 4-6 check the type of distribution. If the distribution is deterministic, the location error is generated and control is returned to the calling routine.

Lines 7-10 check if the distribution is uniform. If it is, the location error is generated and control returns to the calling routine.

Lines 11-13 generate a normal location error and return control to the calling routine.

24. Routine NEW.LOCATION

Description

Routine NEW.LOCATION simulates the FO's estimation of the new location of a cluster at the end of a given time duration. It is used in the adjustment phase and FFE phase. In the adjustment phase, the FO takes two sightings of the centroid of the cluster 20 seconds apart and determines the apparent speed and direction of the cluster. In the FFE phase, the apparent speed and direction determined after the last adjusting rounds are used. In both cases, the routine extrapolates new locations from the last position of the cluster for the time period of the estimated travel. Events CLUSTER.UPDATE and ARTY.IMPACT call routine NEW.LOCATION. It calls routines DIST, POSITION.UPDATE, and FA.TGT.ERROR.

Local Variables

ID.FO - An integer variable representing the FO firing the mission.

ID.MISSION - An integer variable representing the current mission.

TANK - An integer variable representing the vehicle being checked.

DEL.TIME - A real variable equal to the time period over which the location of the cluster has moved.

A - An integer variable representing the type of situation for estimating the location. The variable may take on the following values: 1 - adjustment phase, and 2 - FFE phase.

DISTANCE - A real variable equal to the distance in meters between two sightings.

ERR.1 - ERR.8 - Real variables equal to the sighting error. Since the error was based on a CEP (circular error probable), the error distribution is the same in both the X and Y directions.

X.1 - A real variable equal to the X coordinate of the old location.

Y.1 - A real variable equal to the Y coordinate of the old location.

X.2 - A real variable equal to the X coordinate of the first sighting.

Y.2 - A real variable equal to the Y coordinate of the first sighting.

X.3 - A real variable equal to the X coordinate of the second sighting.

Y.3 - A real variable equal to the Y coordinate of the second sighting.

Coding and Brief Explanation

```
1  ROUTINE NEW.LOCATION(ID.MISSION,DEL.TIME,A)
2  DEFINE ID.MISSION,ID.FO,A AND TANK AS INTEGER VARIABLES
```

```
DEFINE DEL.TIME AS A REAL VARIABLE
DEFINE DISTANCE,ERR.2,ERR.4,ERR.6,ERR.8,X.1,X.3,Y.1,Y.3,ERR.1,
      ERR.3,ERR.5,ERR.7,
X.2,XX,Y.2, AND YY AS REAL VARIABLES
CALL POSITION.UPDATE(ID.MISSION,0)
IF X.CUR4(ID.MISSION) = 0
THEN IF Y.CUR4(ID.MISSION) = 0
LET ERROR.CODE(ID.MISSION) = 6
ALWAYS
IF A = 2
LET X.1 = X.FUTURE.LOC(ID.MISSION)
LET Y.1 = Y.FUTURE.LOC(ID.MISSION)
GO TO NEXT
ELSE
LET ID.FO = FIST(ID.MISSION)
CALL FA.TGT.ERROR(TYPE(ID.FO)) YIELDING ERR.1
CALL FA.TGT.ERROR(TYPE(ID.FO)) YIELDING ERR.2
CALL FA.TGT.ERROR(TYPE(ID.FO)) YIELDING ERR.3
CALL FA.TGT.ERROR(TYPE(ID.FO)) YIELDING ERR.4
LET X.1 = X.CUR4(ID.MISSION) + (ERR.1 + ERR.2)/2
LET Y.1 = Y.CUR4(ID.MISSION) + (ERR.3 + ERR.4)/2
LET X.2 = X.CUR4(ID.MISSION) + SPEED(ID.MISSION) * COS.F
      (DIRECTION(ID.MISSION))
* 20
LET Y.2 = Y.CUR4(ID.MISSION) + SPEED(ID.MISSION) * SIN.F
      (DIRECTION(ID.MISSION))
* 20
```

```
27 CALL FA.TGT.ERROR(TYPE(ID.FO)) YIELDING ERR.5
28 CALL FA.TGT.ERROR(TYPE(ID.FO)) YIELDING ERR.6
29 CALL FA.TGT.ERROR(TYPE(ID.FO)) YIELDING ERR.7
30 CALL FA.TGT.ERROR(TYPE(ID.FO)) YIELDING ERR.8
31 LET X.3 = X.2 + (ERR.5 + ERR.6)/2
32 LET Y.3 = Y.2 + (ERR.7 + ERR.8)/2
33 CALL DIST(X.1, Y.2, X.3, Y.3) YIELDING DISTANCE
34 LET XX = X.3 - X.1
35 LET YY = Y.3 - Y.1
36 LET DIR.APPARENT(ID.MISSION) = ARCTAN.F(YY,XX)
37 LET SPD.APPARENT(ID.MISSION) = DISTANCE/20
38 'NEXT'
39 LET X.FUTURE.LOC(ID.MISSION) = X.1 + SPD.APPARENT(ID.MISSION)
         * COS.F
40 (DIR.APPARENT(ID.MISSION) * DEL.TIME
41 LET Y.FUTURE.LOC(ID.MISSION) = Y.1 + SPD.APPARENT(ID.MISSION)
         * SIN.F
42 (DIR.APPARENT(ID.MISSION) * DEL.TIME
43 RETURN END
```

Lines 1-5 define the routine and the local variables.

Line 6 updates the actual location of the cluster.

Lines 7-10 check if there are any vehicles left alive in the cluster. If none, an error code is established.

Lines 11-15 check if the routine was called from the FFE situation. If it was, the control transfers to line 38.

Line 16 sets the value of ID.FO.

Lines 17-22 call routine FA.TGT.ERROR for errors in sightings and apply the results to the first position.

Lines 23-26 determine the estimated relative position of the cluster as projected 20 seconds from the current time.

Lines 27-32 call routine FA.TGT.ERROR for additional errors in sightings and apply the results to the second position.

Lines 33-37 calculate the mean speed and direction of the cluster based on the two sightings. These values update attributes of ID.MISSION.

Line 38 is a continuation line.

Lines 39-42 extrapolate the future position of the cluster based on the apparent speed and direction of the cluster for the input value of time.

Line 43 returns control to the calling event.

25. Event MRL.IMPACT

Description

Event MRL.IMPACT simulates the effects of a MRL volley in the vicinity of the first TOW team in team B that k-kills a Red tank. The probability of a k-kill, m-kill, and f-kill are calculated and routine ATRIT is called to assess damage. Event MRL.IMPACT is scheduled by event IMPACT. It calls routines LOC, ATRIT and TALLY.HIT.STATE.

Local Variables

ID.BTRY - An integer variable representing the FO.

TYPE.AMMO - An integer variable representing the type of ammunity fired.

RED.2.CONSTANT - An integer variable equal to a conversion factor to be used when calculating the effects on an exposed Dragon gunner.

X.LOC - A real variable equal to the X coordinate of the target.

Y.LOC - A real variable equal to the Y coordinate of the target.

P1 - A real variable equal to the probability of a k-kill.

P2 - A real variable equal to the probability of an m-kill.

P3 - A real variable equal to the probability of an f-kill.

<u>Coding and Brief Explanation</u>

```
UPON MRL.IMPACT(X.LOC,Y.LOC)
DEFINE ID.BTRY,TYPE.AMMO AND RED.2.CONSTANT AS INTEGER VARIABLES
DEFINE P1,P2,P3,X.LOC AND Y.LOC AS REAL VARIABLES
LET TYPE.AMMO = 2
LET ID.BTRY = 11
LET NUM.HE.LEFT(ID.BTRY) = NUM.HE.LEFT(ID.BTRY) - NUM.GUNS(ID.BTRY)
LET NO.MSNS.FIRED(ID.BTRY) = NO.MSNS.FIRED(ID.BTRY) + 1
FOR EACH TANK IN BLUE.ALIVE, DO
CALL LOC(TANK)
IF ABS.F(X.CURRENT(TANK) - X.LOC) LT 200
THEN IF ABS.F(Y.CURRENT(TANK) - Y.LOC) LT 200
IF DEBUG GE 1
PRINT 1 LINE WITH NAME(TANK) THUS
      MRL IMPACT     TANK # **** WAS FIRED ON
ALWAYS
LET RED.2.CONSTANT = 1
IF WPN.TYPE(TANK) = 6
THEN IF DEFNUM(TANK) = 3
LET RED.2.CONSTANT = 7
ALWAYS
LET P1 = ARTY.PK.TABLE(CALIBER(ID.BTRY),
      WPN.TYPE(TANK),TYPE.AMMO,3) * RED.1.CONSTANT
```

```
          * RED.2.CONSTANT
LET P2 = ARTY.PK.TABLE(CALIBER(ID.BTRY),
          WPN.TYPE(TANK),TYPE.AMMO,2) * RED.1.CONSTANT
          * RED.2.CONSTANT
LET P3 = ARTY.PK.TABLE(CALIBER)(ID.BTRY),
          WPN.TYPE(TANK),TYPE.AMMO,1) * RED.1.CONSTANT
          * RED.2.CONSTANT
CALL ATRIT(TANK,TANK,P1,P2,P3,1)
CALL TALLY.HIT.STATE(TANK,DAMAGE.NUM)
IF DAMAGE.NUM = 5
LET ALIVE.DEAD(TANK) = 1
LET HIT.STATE(TANK) = "FA.2"
LET FOE(TANK) = 0
IF FA(TANK) NE 0
SCHEDULE AN ALT.FO(TANK) IN 100 UNITS
ALWAYS
ALWAYS
START NEW LINE
WRITE
          NCASE, GUNTUBE,     TYPE.AMMO,          NAME(TANK),
CALIBER(ID.BTRY),TIME.V,
          X.CURRENT(TANK),Y.CURRENT(TANK),Z.CURRENT(TANK),SPD(TANK),
DEFNUM(TANK),
FKILL(TANK), MFKILL(TANK),
          KKILL(TANK),K.HIT(TANK),FIRED.AT(TANK)
AND NUM.HIT(TANK) AS
I 2,I 3, "ARTY",S 8,
          3 I 4, I 5, S 20, S 7, 3 I 5, D(4,1), I 3, S 11,
```

```
50         S 1, A 4, 2 D(5,2), 2 I 2, 5 I 3
51  ALWAYS
52  LOOP
53  RETURN END
```

Lines 1-3 define the event and the local variables.

Line 4 defines the type of ammunition.

Line 5 defines the firing unit.

Lines 6-7 update the ammunition count and the number of missions fired.

Lines 8-11 screen for vehicles that lie within the MRL volley's effects pattern.

Lines 12-13 are a print option.

Lines 15-19 define a conversion factor for use in determining effects on an exposed Dragon gunner.

Lines 20-28 calculate the probability of different levels of damage.

Lines 29-30 assess damage and call bookkeeping routine.

Lines 31-34 update attributes if the Blue vehicle is killed.

Lines 35-37 schedule a search for substitute FO if FO k-killed.

Lines 39-50 print out damage information.

Line 53 returns control to the system timer.

26. Routine PREPLANNED

Description

Routine PREPLANNED generates a table of rectangle locations for use in the Red planned fires. The values are stored in the array called RED.PLANNED FIRES with three subscripts. The first subscript indicatesthe iteration value. The second subscript indicates the following: 1- X coordinate of the upper left corner of the rectangle, 2- Y coordinate of the upper left corner of the

rectangle, 3- X coordinate of the lower right corner of the rectangle, and 4- Y coordinate of the lower right corner of the rectangle. The third subscript indicates the Red battery identification number, starting with the value of 1 for the first Red battery created. Routine PREPLANNED is called by routine FA.1.MAIN.

Local Variables

A - An integer variable representing the number of Red batteries whose fires are being planned.

B - An integer variable representing the value of the uppermost X coordinate in the rectangle.

C - An integer variable representing the change in X coordinate per shift in fires.

D - An integer variable representing the uppermost Y coordinate in the rectangle.

E - An integer variable representing the change in Y coordinate per shift in fires.

F - An integer variable representing the value of the last Red battery whose fires were planned prior to this routine being called.

I - An integer variable representing the row number of the target in array.

J - An integer variable used to construct the coordinate array.

K - An integer variable used to construct the coordinate array.

L - An integer variable representing the third dimension value in the array used to construct the coordinates.

Coding and Brief Explanation

```
1  ROUTINE PREPLANNED(A,B,C,D,E,F)
2  DEFINE A,B,C,D,E,F,I,J,K AND L AS INTEGER VARIABLES
3  LET J = 0
4  LET K = 0
5  LET L = F+1 TO A+F, DO
6  FOR I=1 TO 29 BY 2, DO
7  LET J=J+1
8  LET RED.PLANNED.FIRES(I,1,L) = B +((L-F-1)*6)
9  LET RED.PLANNED.FIRES(I,2,L) = D -J+1
10 LET RED.PLANNED.FIRES(I,3,L) = B + C + ((L-F-1)*6)
11 LET RED.PLANNED.FIRES(I,4,L) = RED.PLANNED.FIRES(I,2,L) -E
12 LOOP
13 FOR I = 2 TO 30 BY 2, DO
14 LET K = K+1
15 LET RED.PLANNED.FIRES(I,1,L) = B + C + ((L-F-1)*6)
16 LET RED.PLANNED.FIRES(I,2,L) = D -K+1
17 LET RED.PLANNED.FIRES(I,3,L) = B + (2*C) + ((L-F-1)*6)
18 LET RED.PLANNED.FIRES(I,4,L) = RED.PLANNED.FIRES(I,2,L) -E
19 LOOP
20 LET J = 0
21 LET K = 0
22 REPEAT
23 RETURN END
```

Lines 1-2 define the routine and local variables.

Lines 3-22 construct the array of coordinate values.

Line 23 returns control to the calling routine.

27. Routine POSITION.UPDATE

Description

Routine POSITION.UPDATE updates the position of the centroid of a given cluster. First, it updates the location of all TANKS in the cluster and calculates the mean X coordinate, Y coordinate, and speed. This event is called only during the adjustment phase of a mission. Event POSITION.UPDATE is called by routines ERROR and NEW.LOCATION.

Local Variables

ID.MISSION - An integer variable representing the mission that is being fired.

ID.FO - An integer variable representing the FO firing on the cluster.

TANK - An integer variable representing the vehicle being processed.

X.BAR - A real variable equal to the mean of the X coordinates.

Y.BAR - A real variable equal to the mean of the Y coordinates.

SPD.BAR - A real variable equal to the mean of the speeds.

Coding and Brief Explanation

```
ROUTINE POSITION.UPDATE(ID.MISSION,A)
DEFINE ID.FO,ID.MISSION,TANK AND A AS INTEGER VARIABLES
DEFINE Y.BAR,X.BAR AND SPD.BAR AS REAL VARIABLES
LET ID.FO = FIST(ID.MISSION)
FOR EACH TANK IN RED.ALIVE, DO
IF C.NUMBER.ARRAY(NAME(TANK)-B.NUM.ALIVE, ID.FO,STATE4(ID.FO)) =
NO.CLUSTER(ID.MISSION)
CALL LOC(TANK)
COMPUTE X.BAR AS THE MEAN OF X.CURRENT(TANK)
```

```
10 COMPUTE Y.BAR AS THE MEAN OF Y.CURRENT(TANK)
11 COMPUTE SPD.BAR AS THE MEAN OF SPD(TANK)
12 ALWAYS
13 LOOP
14 LET X.CUR4(ID.MISSION) = X.BAR
15 LET Y.CUR4(ID.MISSION) = Y.BAR
16 LET SPEED(ID.MISSION) = SPD.BAR
17 RETURN END
```

Lines 1-3 define the routine and the local variables.

Line 4 defines the variable ID.FO.

Lines 5-7 search for each vehicle whose cluster identification in the array called C.NUMBER.ARRAY equals that of the cluster being fired upon.

Line 8 updates the vehicle location.

Lines 9-12 compute the appropriate mean of all live vehicles in the cluster.

Lines 14-16 update the appropriate attributes of the cluster.

Line 17 returns control to the calling routine.

28. Event ALT.FO

Description

Event ALT.FO attempts to replace an FO who has had his vehicle k-killed. The event checks each platoon leader in the FO's company. The first live platoon leader is given the FO's duties. If none are alive, the duties are not passed on. This event is scheduled by event IMPACT, event RED.PLANNED FIRES, and event MRL.IMPACT.

Local Variables

A - An integer variable representing the vehicle in which the FO was riding when he was killed.

M - An integer variable used to cut off the search through the set BLUE.ALIVE after first live FO replacement is found.

Coding and Brief Explanation

```
1  UPON ALT.FO(A)
2  DEFINE A AND M AS INTEGER VARIABLES
3  FOR EACH TANK IN BLUE.ALIVE WITH CO(TANK) = CO(A)
4  AND TANK = PLTLDR(TANK) UNTIL M = 1, DO
5  LET FA(TANK) = FA(A)
6  LET TYPE(FA(A)) = 2
7  LET POINTING.TO(FA(A)) = POINTER(TANK)
8  LET M = 1
9  LOOP
10 RETURN END
```

Lines 1-2 define the event and the local variables.

Lines 3-4 search for the first live platoon leader in the FO's company.

Lines 5-7 transfer the old FO's attributes to the new FO.

Line 10 returns control to the system timer.

29. Event SHUT.OFF

Description

Event SHUT.OFF simulates the termination of the Red planned fires as the Red forces close onto their objectives. A global variable, RED.OFF, is given the value of one when this event is processed. Event SHUT.OFF is scheduled by event STEP.TIME.

Coding and Brief Explanation

```
UPON SHUT.OFF
LET RED.OFF = 1
RETURN END
```

Line 1 defines the event.

Line 2 changes the value of the global variable RED.OFF.

Line 3 returns control to the system timer.

C. STAR ROUTINES AND EVENTS THAT REQUIRED MODIFICATION

The following modifications to the STAR model are required to facilitate the addition of the FA module. Since the STAR model is very dynamic in nature, the line numbers and actual code alterations are correct only for the current version of STAR. The reason for the modification is specific enough to allow the user to make the appropriate modification to the user's version of the STAR model. Only local variables that are required for the modification are described.

1. PREAMBLE

The integer attribute, FA, was added to the attribute list of the temporary entity, TANK. It indicated that the vehicle belonged to the FO; otherwise its value was zero. A number of the following modifications check this attribute when preventing the FO from performing inappropriate activities.

2. Routine RED.CREATE

After Red units are created, the array called C.NUMBER.ARRAY is dimensioned dynamically. The dimension increases as the additional forces are created.

Coding and Brief Explanation

```
37  IF TIME.V LT 120 JUMP AHEAD ELSE
38  RELEASE C.NUMBER.ARRAY(*,*,*)
39  HERE
40  RESERVE C.NUMBER.ARRAY(*,*,*) AS BC.COUNT BY N.FO BY
41  MAX.NUMBER.OF.MISSIONS.PER.FO
```

Lines 37-39 release the array only if it had been previously reserved. (Time 120 is the second time Red forces are created. This was specific for this given scenario and set of runs.)

Lines 40-41 reserve the array according to the total Red forces that have been created.

3. <u>Routine BASIC.LOAD</u>

Routine BASIC.LOAD provides ammunition to the Blue forces. When the FO's vehicle (an ITV with WPN.TYPE(TANK) = 4) is being armed, a filter was applied that skipped the ammunition supply and changed the value of the variable OP.RNG to equal the FO's maximum range for locating targets.

<u>Local Variable</u>

A - An integer variable representing the FO's vehicle.

<u>Code</u>

```
IF NAME(A) = 3 OR NAME(A) = 78 OR NAME(A) = 79
     LET OP.RANG(A) = 3800 JUMP AHEAD ELSE
HERE
```

4. <u>Event TARGET.SELECT</u>

Event TARGET.SELECT simulated the procedures used in selecting a direct fire target. A filter was added that prevented the FO from selecting direct fire targets (by returning control to the system timer).

5. <u>Event IMPACT</u>

Routine IMPACT simulates a direct fire round arriving at the target. Event ALT.FO is scheduled if the FO's vehicle is k-killed. When the first Red tank is k-killed by a TOW fired from team B's position, a MRL.IMPACT is scheduled.

<u>Local Variables</u>

ID - An integer variable representing the vehicle that was fired at.

A - An integer variable representing the vehicle that did the firing.

Coding and Brief Explanation

```
35 IF FA(ID) NE 0
36 SCHEDULE AN ALT.FO(ID) IN 100 UNITS ALWAYS
37 HERE
38 CALL TALLY.HIT.STATE(ID,DAMAGE.NUM)
39 ''DECISION RULE FOR REQUESTING MRL
40 IF CO(A) = 2 AND (WPN.TYPE(A)=3 OR WPN.TYPE(A)=4) AND ALIVE.DEAD(ID)
41 PROJO(A)=1 AND WPN.TYPE(ID)=7
42 THEN IF TOW.KOUNT = 0
43 LET TOW.KOUNT = 1
44 SCHEDULE AN MRL.IMPACT(X.CURRENT(A),Y.CURRENT(A)) IN 300 UNITS
45 ALWAYS
```

Lines 35-36 schedule the search for the FO's replacement.

Lines 40-41 check whether this impact was a TOW fired from team B whether it killed a Red tank.

Lines 42-45 schedule a volley from the MRL battery the first time that a TOW was fired from team B and killed a Red tank.

6. Event STOP.SIMULATION

Event STOP.SIMULATION prints the final statistics of the battle. Attributes of all the FA entities were printed.

Code

```
START NEW PAGE
LIST ATTRIBUTES OF EVERY FO
LIST ATTRIBUTES OF EVERY BATTERY
LIST ATTRIBUTES OF EVERY FDC
LIST ATTRIBUTES OF EVERY MISSION IN MSN.QUEUE
```

```
25  LIST ATTRIBUTES OF EVERY MISSION IN HOLDING.MSNS
```

7. Routine LIST.UPDATE

Routine LIST.UPDATE updates the target list of a vehicle and schedules the selection of a target. A filter was added that insured that the FO did not select a direct fire target (by returning control to the calling event).

8. Event STEP.TIME

Event STEP.TIME schedules detections for Red and Blue elements. It releases a vehicle's target list if that vehicle is in or going into full defilade (DEFNUM = 1), simulating the tank commander's loss of line of sight to his targets. If the FO's vehicle is in defilade, it is reasonable to assume that the FO would still be involved in target acquisition and not forget his targets. A filter was added that ensured that the FO's target list was not released. Another modification was added that checked the distance between the two vehicles to determine if the opposing vehicle is within 1500 meters. If so, the Red planned fires were scheduled to be terminated in 100 seconds.

Local Variables

A - An integer variable representing the vehicle being checked.

R - A real variable equal to the range between the two vehicles (one Red and one Blue).

Coding and Brief Explanation

```
12  IF FA(A) NE (0) GO TO LIST1 ELSE
20  'LIST1'
39  IF R LE 1500 SCHEDULE A SHUT.OFF IN 100 UNITS ALWAYS
```

Line 12 transfers control around code that releases the vehicles list when in full defilade.

Line 39 checks range between opposing vehicles. If less than 1500, a termination of Red fires is scheduled.

9. Event DETECT

Event DETECT simulates the detection of a vehicle. If a detection is scheduled and the detected or detecting vehicle was in full defilade, the detection was not allowed to happen. A filter was added to ensure that the FO continued to detect targets even while in full defilade (by transferring control around the releasing code). If the FO's vehicle is in defilade, it is reasonable to assume that the FO would still be involved in target acquisition. One of the members of the fire support team would dismount the vehicle and move to high ground in order to be able to observe.

VI. A SIMULATED FIRE MISSION

The direct support artillery firing unit provides an excellent example of the Field Artillery system and how it operates. This chapter presents one representative mission from a production run of the current STAR model with the FA module added. A mission fired by a four gun firing unit of a 155mm M109 howitzer battery (armored division) was simulated. Figures 5-17 demonstrate the effects of the mission along with the detailed computer output available with the DEBUG option set equal to 5. The output follows each phase of the mission, giving details that were used to make the plots, and the time of each event's occurrence. Battle time is shown in seconds from the beginning of the simulation.

The battery was in direct support of a battalion-sized task force that had three FOs. The FO with company team B considered all vehicles that he had detected prior to clustering. Figure 5 shows the plot of these vehicles. The FO then clustered all the vehicles that he had detected as of battle time 922. Figure 6 shows the plot of the FO's position and the vehicles after they were placed in clusters. Figure 7 is the computer output that gives the following information: Cluster number (as indicated in Figure 6), priority, number of vehicles in cluster, the X and Y coordinates of the cluster centroid, the speed and direction of the cluster centroid, and the X and Y coordinates of the first vehicle placed in the cluster. The FO decided to adjust fire on cluster number 1 (the cluster with the highest priority) and to use DPICM ammunition. The FO called for fire over his radio to the second firing unit FDC. His call for fire included the location of the target, its description, and requested a method of engagement and a method of fire and control. The FDO accepted the FO's request and

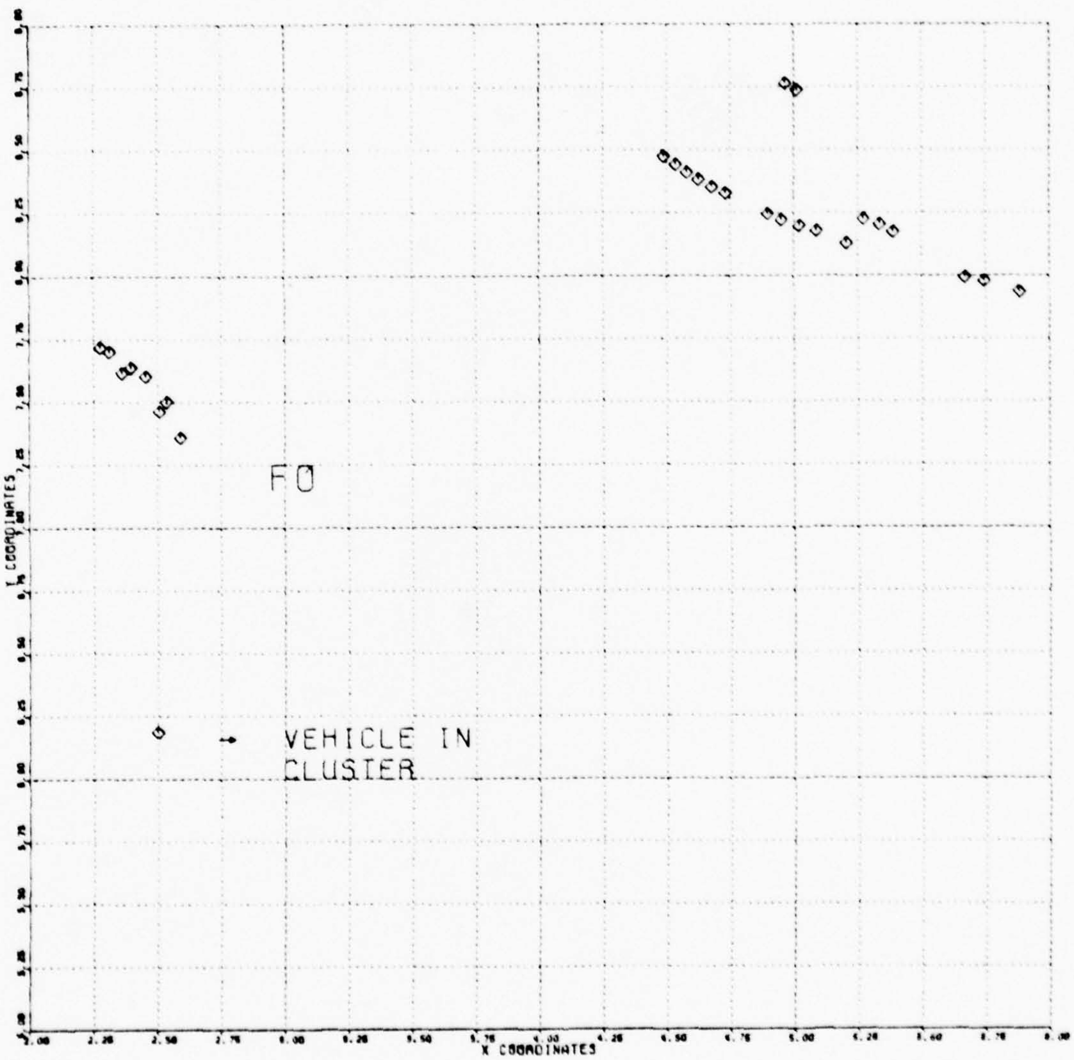


Figure 5. Vehicles that FO has detected.

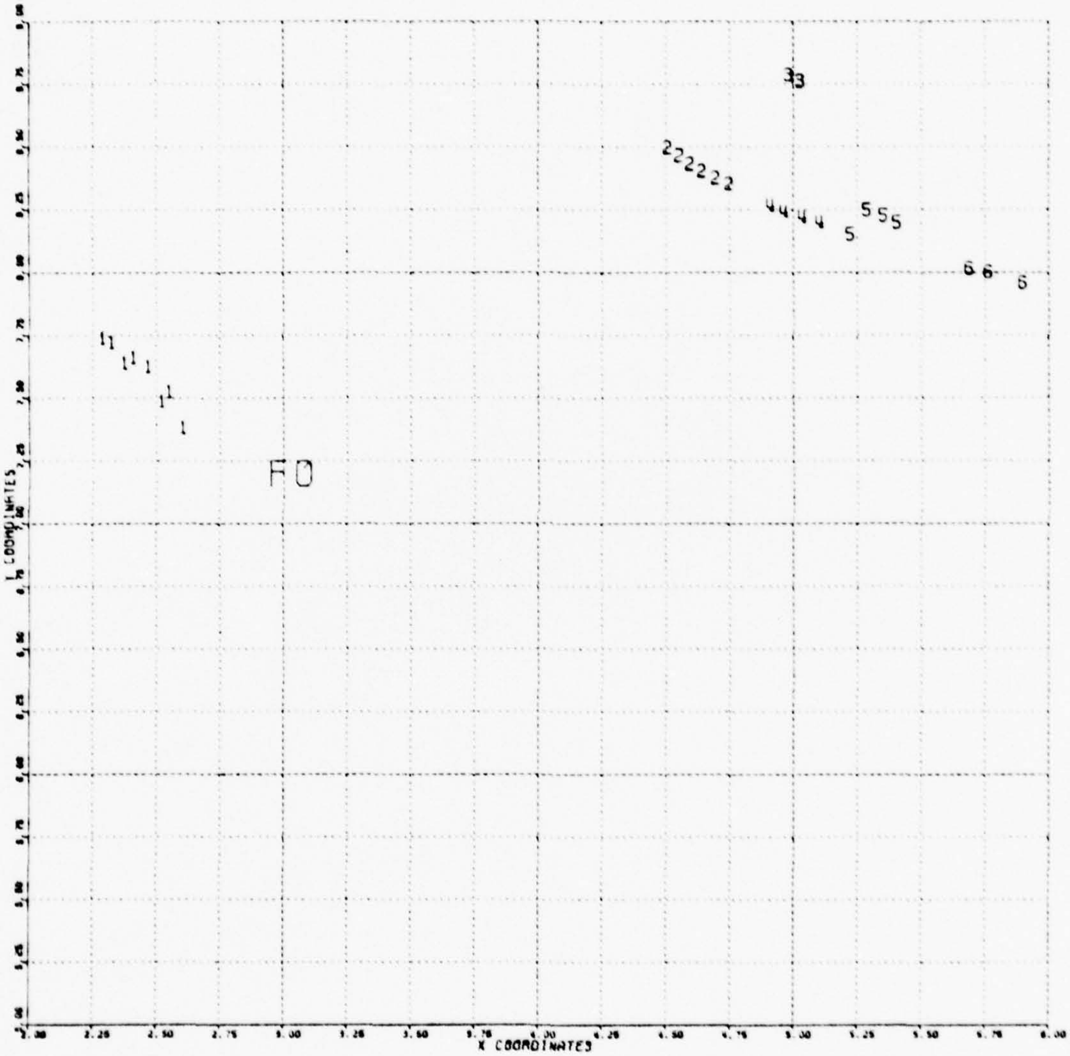


Figure 6. Forward observer's cluster results.

FO 3 CLUSTERING TIME = 922.2

CLUSTER ARRAY FOR FO# 3

NUM	#	PRI	X	Y	SPD	DIR	XFIR	YFIR
1	8.	5.9	2428.	7576.	3.	-2.	2506.	7464.
2	6.	2.9	4606.	8398.	6.	-3.	4579.	8412.
3	2.	0.	4988.	8753.	6.	-2.	4965.	8765.
4	4.	1.7	4990.	8211.	6.	-3.	4896.	8246.
5	4.	1.6	5301.	8184.	6.	-3.	5206.	8130.
6	3.	1.0	5766.	7969.	6.	-3.	5670.	7994.
7	0.	0.	0.	0.	0.	0.	0.	0.
8	0.	0.	0.	0.	0.	0.	0.	0.
9	0.	0.	0.	0.	0.	0.	0.	0.
10	0.	0.	0.	0.	0.	0.	0.	0.
11	0.	0.	0.	0.	0.	0.	0.	0.
12	0.	0.	0.	0.	0.	0.	0.	0.
13	0.	0.	0.	0.	0.	0.	0.	0.
14	0.	0.	0.	0.	0.	0.	0.	0.
15	0.	0.	0.	0.	0.	0.	0.	0.
16	0.	0.	0.	0.	0.	0.	0.	0.
17	0.	0.	0.	0.	0.	0.	0.	0.
18	0.	0.	0.	0.	0.	0.	0.	0.
19	0.	0.	0.	0.	0.	0.	0.	0.
20	0.	0.	0.	0.	0.	0.	0.	0.

FO 3 COMPUTING MSN MSN 10
SPC 3. X 2428. Y 7576. DIR -1.6
FO RANGE TO TGT 674.

Figure 7. Cluster information.

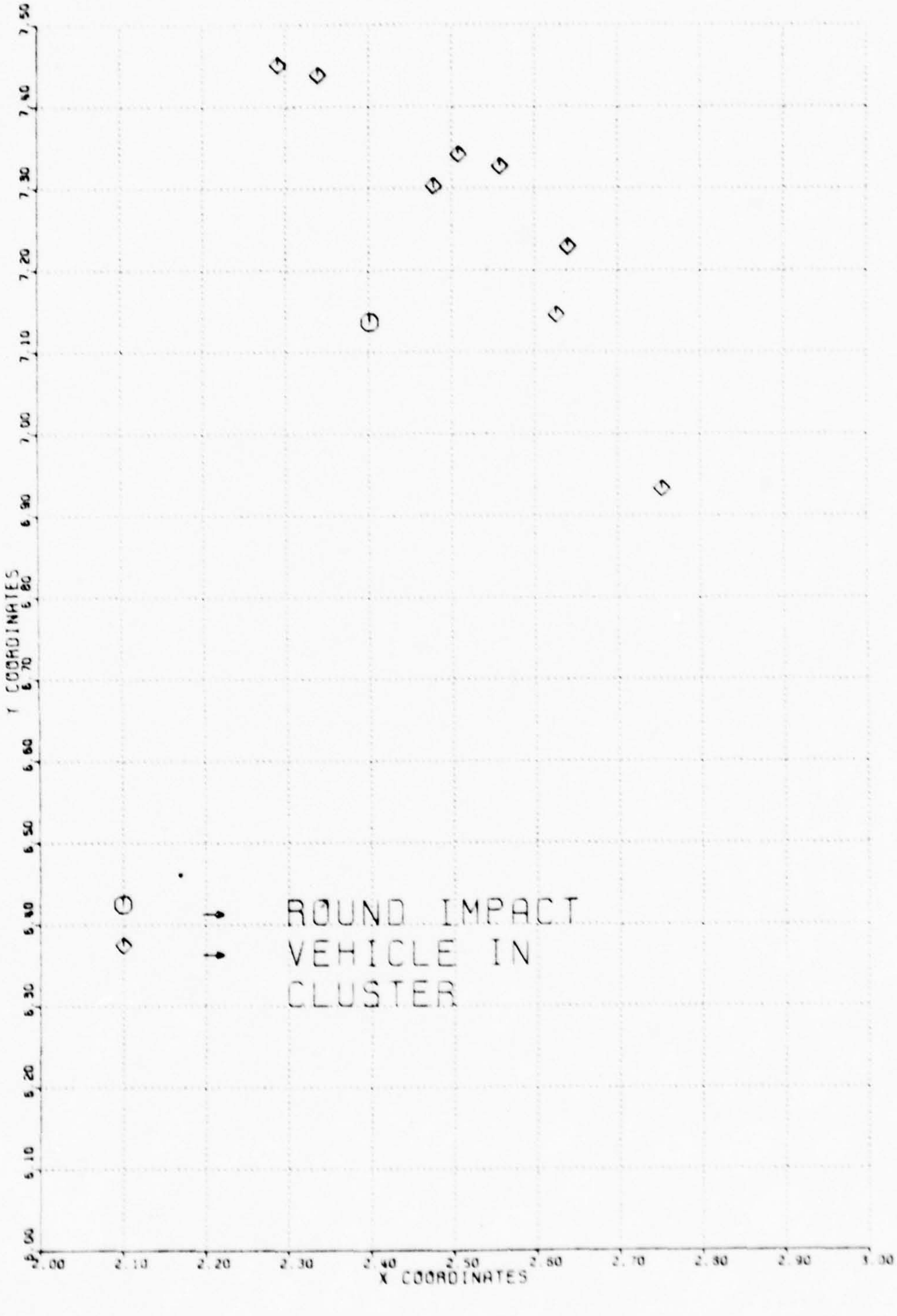


Figure 8. First adjusting round.

```
FO 3  ( 10 )        FDC PROCESSING        TIME=   951.
                    FDC 1    0 MSNS  1 BTRY  0  QUEUE
                    FDC 2    0 MSNS  0 BTRY  0  QUEUE
                    FDC 3    0 MSNS  0 BTRY  0  QUEUE

FO 3  ( 10 )        CHECKING GUNS         TIME=   960.
                    BTRY 2  GUNS 0

FO 3  ( 10 )        FIRING GUNS           TIME=   983.
                    BTRY 2 VOLLEY  0

FO 3  ( 10 )        ARTY IMPACT           TIME=  1006.
                    BTRY 2 ADJ RD 1 IS  182. METERS FROM TGT

FO 3  ( 10 )        ENTERING FFE          TIME=  1006.
```

Figure 9. Computer Output: Time 951-1006.

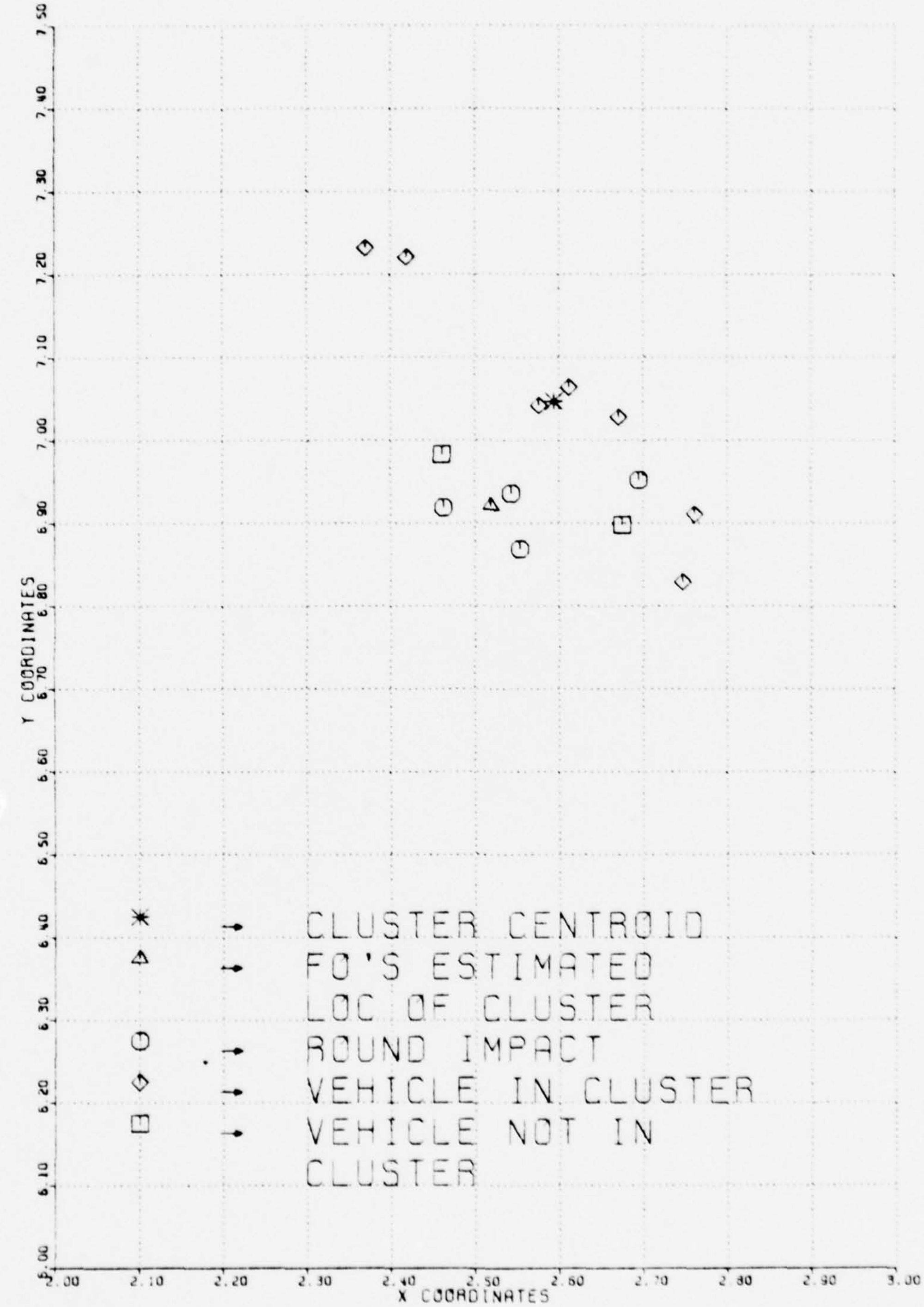


Figure 10. First volley in FFE phase.

```
FO 3  ( 10 )        FIRING GUNS           TIME= 1040.
                    BTRY 2 VOLLEY  1

FO 3  ( 10 )        ARTY IMPACT           TIME= 1063.
                    BTRY 2  VOLLEYS LEFT   3

MISSION NUMBER                         10
TIME                                 1063.
CLUSTER CENTER                       2594.       7047.
FUTURE LOC                           2519.       6921.
MPI LOCATION                         2562.       6909.
LOCATION OF RD#   1                  2695.       6952.
LOCATION OF RD#   2                  2553.       6869.
LOCATION OF RD#   3                  2543.       6936.
LOCATION OF RD#   4                  2463.       6919.
NUMBER OF TANKS IN RECT                 7
LOCATION OF TANK #   86              2461.       6983.
LOCATION OF TANK #   98              2675.       6898.
LOCATION OF TANK #   99              2747.       6829.
LOCATION OF TANK #  123              2576.       7042.
LOCATION OF TANK #  132              2612.       7064.
LOCATION OF TANK #  133              2671.       7028.
LOCATION OF TANK #  134              2761.       6911.
NUMBER OF TANKS IN CLUSTER              7
LOCATION OF TANK #   99              2747.       6829.
LOCATION OF TANK #  120              2370.       7232.
LOCATION OF TANK #  121              2419.       7221.
LOCATION OF TANK #  123              2576.       7042.
LOCATION OF TANK #  132              2612.       7064.
LOCATION OF TANK #  133              2671.       7028.
LOCATION OF TANK #  134              2761.       6911.

                     ..... TANK #   86 WAS FIRED ON    .....
                RD#  4 LANDED  22. METERS FROM TANK#  86
                     ..... TANK #   98 WAS FIRED ON    .....
                     ..... TANK #   99 WAS FIRED ON    .....
                     ..... TANK #  123 WAS FIRED ON    .....
                     ..... TANK #  132 WAS FIRED ON    .....
                     ..... TANK #  133 WAS FIRED ON    .....
                     ..... TANK #  134 WAS FIRED ON    .....

FO 3  ( 10 )        FIRING GUNS           TIME= 1064.
                    BTRY 2 VOLLEY  2
```

Figure 11. Computer Output: Time 1040-1064

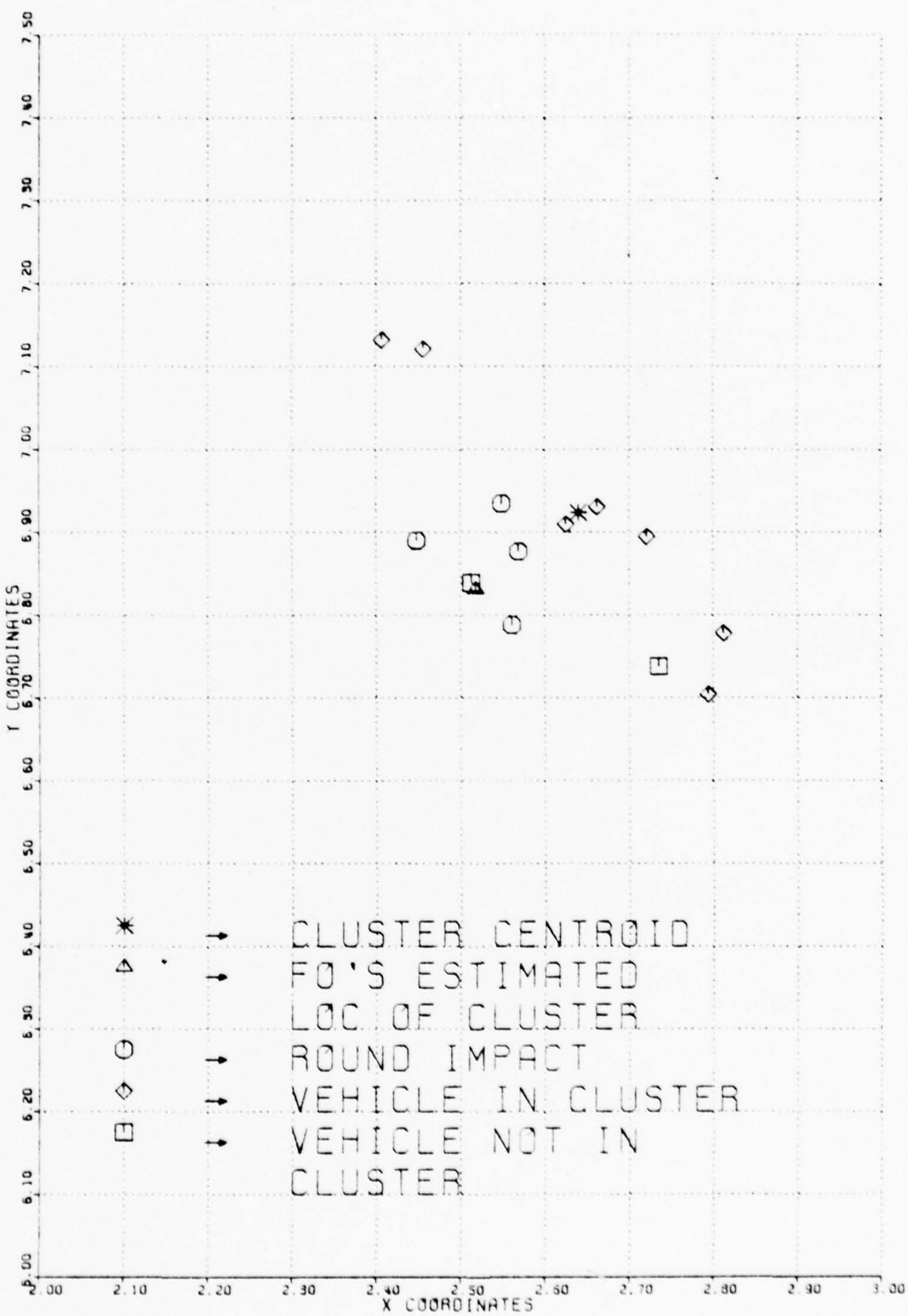


Figure 12. Second volley in FFE phase.

```
FO 3  ( 10 )       FIRING GUNS           TIME= 1085.
                   BTRY 2 VOLLEY  3

FO 3  ( 10 )       ARTY IMPACT           TIME= 1087.
                   BTRY 2  VOLLEYS LEFT   2

MISSION NUMBER                          10
TIME                                  1087.
CLUSTER CENTER                        2640.       6924.
FUTURE LOC                            2518.       6831.
MPI LOCATION                          2547.       6893.
LOCATION OF RD#   1                   2561.       6788.
LOCATION OF RD#   2                   2569.       6877.
LOCATION OF RD#   3                   2549.       6935.
LOCATION OF RD#   4                   2448.       6890.
NUMBER OF TANKS IN RECT                  9
LOCATION OF TANK #   86               2513.       6839.
LOCATION OF TANK #   98               2735.       6738.
LOCATION OF TANK #   99               2794.       6704.
LOCATION OF TANK #  120               2407.       7132.
LOCATION OF TANK #  121               2456.       7121.
LOCATION OF TANK #  123               2625.       6910.
LOCATION OF TANK #  132               2662.       6931.
LOCATION OF TANK #  133               2721.       6895.
LOCATION OF TANK #  134               2812.       6778.
NUMBER OF TANKS IN CLUSTER               7
LOCATION OF TANK #   99               2794.       6704.
LOCATION OF TANK #  120               2407.       7132.
LOCATION OF TANK #  121               2456.       7121.
LOCATION OF TANK #  123               2625.       6910.
LOCATION OF TANK #  132               2662.       6931.
LOCATION OF TANK #  133               2721.       6895.
LOCATION OF TANK #  134               2821.       6778.

                    ..... TANK #   86 WAS FIRED ON    .....
               RD#  1 LANDED  37. METERS FROM TANK#  86
                    ..... TANK #   98 WAS FIRED ON    .....
                    ..... TANK #   99 WAS FIRED ON    .....
                    ..... TANK #  120 WAS FIRED ON    .....
                    ..... TANK #  121 WAS FIRED ON    .....
                    ..... TANK #  123 WAS FIRED ON    .....
                    ..... TANK #  132 WAS FIRED ON    .....
                    ..... TANK #  133 WAS FIRED ON    .....
                    ..... TANK #  134 WAS FIRED ON    .....
```

Figure 13. Computer Output: Time 1085-1087.

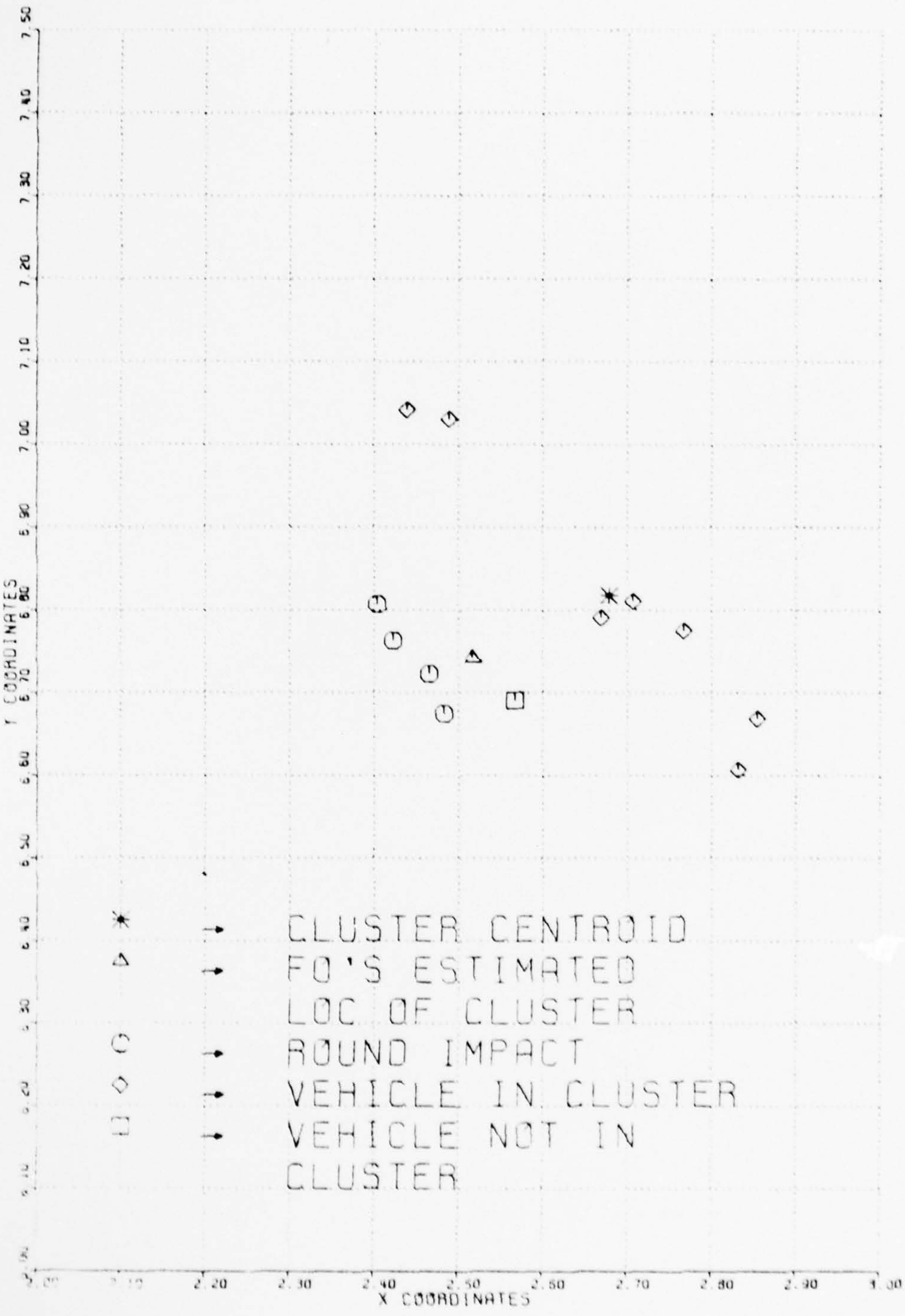


Figure 14. Third volley in FFE phase.

```
FO 3  ( 10 )        ARTY IMPACT            TIME= 1108.
                    BTRY 2  VOLLEYS LEFT   1

MISSION NUMBER                            10
TIME                                    1108.
CLUSTER CENTER                          2679.       6818.
FUTURE LOC                              2517.       6742.
MPI LOCATION                            2466.       6767.
LOCATION OF RD#   1                     2483.       6675.
LOCATION OF RD#   2                     2465.       6723.
LOCATION OF RD#   3                     2423.       6763.
LOCATION OF RD#   4                     2404.       6807.
NUMBER OF TANKS IN RECT                    5
LOCATION OF TANK #   86                 2567.       6691.
LOCATION OF TANK #  120                 2439.       7042.
LOCATION OF TANK #  121                 2489.       7031.
LOCATION OF TANK #  123                 2669.       6791.
LOCATION OF TANK #  132                 2707.       6811.
NUMBER OF TANKS IN CLUSTER                 7
LOCATION OF TANK #   99                 2831.       6607.
LOCATION OF TANK #  120                 2439.       7042.
LOCATION OF TANK #  121                 2489.       7031.
LOCATION OF TANK #  123                 2669.       6791.
LOCATION OF TANK #  132                 2707.       6811.
LOCATION OF TANK #  133                 2767.       6775.
LOCATION OF TANK #  134                 2854.       6669.

                     ..... TANK #   86 WAS FIRED ON    .....
                     ..... TANK #  120 WAS FIRED ON    .....
                     ..... TANK #  123 WAS FIRED ON    .....
                     ..... TANK #  132 WAS FIRED ON    .....

FO #  ( 10 )        FIRING GUNS            TIME= 1109.
                    BTRY 2 VOLLEY  4
```

Figure 15. Computer Output: Time 1108-1109.

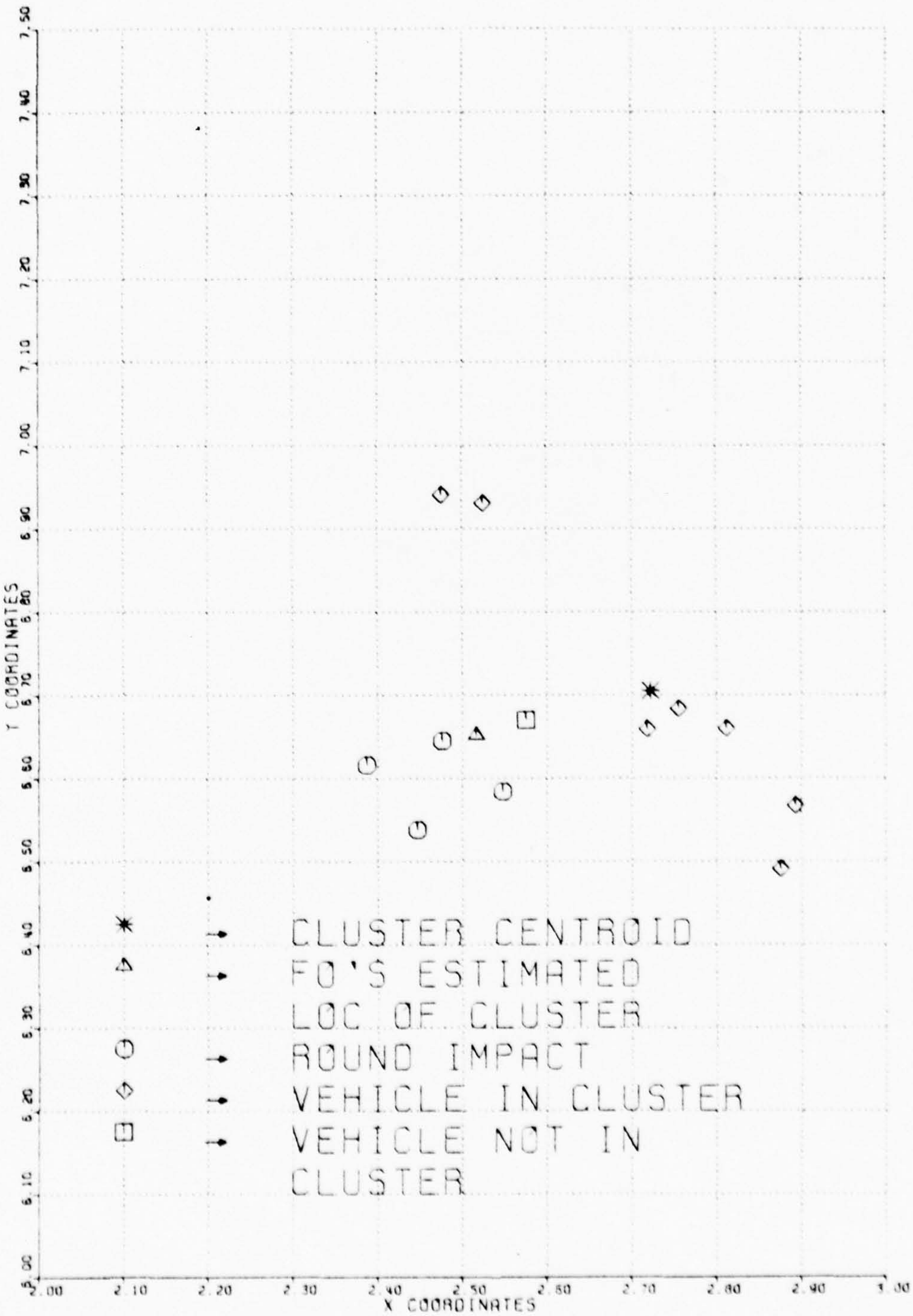


Figure 16. Fourth volley in FFE phase.

```
FO 3  ( 10 )       ARTY IMPACT         TIME= 1132.
                   BTRY 2  VOLLEYS LEFT  0

MISSION NUMBER                        10
TIME                                1132.
CLUSTER CENTER                      2722.      6705.
FUTURE LOC                          2516.      6652.
MPI LOCATION                        2462.      6596.
LOCATION OF RD#   1                 2548.      6584.
LOCATION OF RD#   2                 2447.      6538.
LOCATION OF RD#   3                 2476.      6645.
LOCATION OF RD#   4                 2387.      6616.
NUMBER OF TANKS IN RECT                3
LOCATION OF TANK #   86             2575.      6670.
LOCATION OF TANK #  123             2718.      6659.
LOCATION OF TANK #  132             2755.      6684.
NUMBER OF TANKS IN CLUSTER             7
LOCATION OF TANK #   99             2875.      6491.
LOCATION OF TANK #  120             2476.      6941.
LOCATION OF TANK #  121             2525.      6931.
LOCATION OF TANK #  123             2718.      6659.
LOCATION OF TANK #  132             2755.      6684.
LOCATION OF TANK #  133             2811.      6660.
LOCATION OF TANK #  134             2893.      6567.

                    ..... TANK #   86 WAS FIRED ON    .....
                    ..... TANK #  123 WAS FIRED ON    .....
                    ..... TANK #  132 WAS FIRED ON    .....

FO 3  ( 10  )      END OF MISSION      TIME= 1142.
                   BTRY 2
```

Figure 17. Computer Output: Time 1132-1142.

the battery computer system (BCS) provided a solution to the ballistic problem of getting the rounds to land on the target. The firing data for the first adjusting round was sent to a howitzer section. The section chief positioned his howitzer for direction and elevation in accordance with the firing data. The specified ammunition, DPICM, was prepared, loaded, and fired at battle time 983. The FO located the impact point of the adjusting round (Figure 8). Since the round landed within 300 meters of the cluster centroid (radial error to 182 meters as indicated on Figure 9), the FO entered the fire for effect phase FFE) when he sent his first corrections to the FDC (at battle time 1006). All weapons in the firing unit then fired at their maximum rate of fire for four volleys. Figures 10 and 11 give the plot of the first volley and the computer output for battle time 1040-1064. Figures 12-13 give the plot of the second volley and the computer output for battle time 1085-1087. Figures 14-15 give the plot of the third volley and the computer output for battle time 1108-1109. Figures 16-17 give the plot of the fourth volley and the computer output for battle time 1132-1142. The FO observed the effects and ended the mission at battle time 1142 (as indicated on Figure 17).

VII. EXTENSIONS OF THE BATTALION MODEL

Two major tasks in any modeling effort are deciding which functions to model and determining the subsequent level of detail. These decisions are a function of the intended application of the model and the resources available. When new applications arise and additional resources are available, the model can be modified and extended as desired. A number of enhancements to the Field Artillery model merit consideration and are discussed in this chapter.

A. TARGET ACQUISITION

1) A very simple cluster priority scheme was used. A more elaborate weighting scheme could be developed that should consider, at the least, the type of vehicle, the current status of the battle, and a shifting of priority for range based on range bands

2) The FO used adjustment procedures that estimated the time duration between his call for fire and the round/volley impact. An alternative method would be to use time on target (TOT) procedures, i.e. the FO tells the FDC the exact time that he wants the round/volley to land. One variable in the extrapolation would be eliminated at the expense of increasing the duration of the mission.

3) Another method of observed fire is to fire for effect without prior adjustment. The resulting miss distance might prove to be acceptable in light of the quicker servicing of the target and more efficient utilization of a scarce resource (the firing unit).

4) Targets that are acquired from assets not organic to the battery level could be externally generated after a suitable delay that simulates

the time of processing and the communication of the information to the firing unit.

B. COMMAND AND CONTROL

The current model does not consider the effectiveness of munitions and the massing of fires in determining the number of volleys to fire in the FFE phase. An algorithm could be devised that would determine the best method of engagement (e.g., how many units to fire and how many volleys to fire) to achieve a desired level of effectiveness (commander's guidance).

C. AMMUNITION

1) The use of cannon launched guided projectiles (CLGP) was not modeled. An excellent extension would be to model such a capability, using a two gun platoon from one of the 155mm firing units. External to the model, visibility diagrams would be made of the area on which the Red units would advance. The areas of uninterrupted visibility would be used to select proper target areas (called footprints) in which the FO may attack using CLGP. Before he clusters, the FO would consider the future areas in which he would have line of sight of enemy vehicles for at least 15 seconds. After choosing the cluster with the highest priority of those which would move through one of the footprints, the FO would request that some number of CLGPs be fired into that footprint.

2) The use of Field Artillery delivered mines would assist in disrupting enemy movement and/or destroying vehicles. The effects pattern would be predetermined. When the volley impacts, the effectiveness can be determined from the mean point of impact of the volley. As enemy vehicles enter the area, an assessment of the damage due to mines can be made.

3) Artillery delivered smoke can be fired in support of both offensive and defensive operations. A smoke module for the STAR model is under current development. When a smoke round impacts, the smoke module would be called to generate the proper obstruction and restriction of visibility for some prescribed time.

D. DAMAGE ASSESSMENT

Suppression was not modeled because there is no operational definition that is generally accepted and therefore no data base exists. As an exploratory alternative, a parametric analysis of suppression effects could be accomplished in the following manner. Suppression could be arbitrarily defined (for armored targets) with the following characteristics: 1) the time period of suppression would be from impact of the first round until 5 seconds after the last round, and 2) tank crews would be prevented from engaging with direct fire and would lose all detections during the period of the suppression. The lethal area for suppression would be a constant multiplied times the appropriate k-kill lethal area. There would be a radius of suppression effects within which the time of suppression would be a function of the miss distance. As an individual parameters (lethal area of suppression and effects on the crew) and the time function are varied, the sensitivity of the model to these parameters could be examined.

VIII. THE BRIGADE MODEL

This chapter outlines the general requirements for a Field Artillery model to support the brigade level version of STAR. The major areas addressed (in addition to those found in the FA battalion model) are resource allocation, counterfires, fire support coordination, additional target acquisition capabilities, and the movement of firing units. Only the Blue organization, equipment, and tactics will be discussed. The Red force can be modeled in a manner similar to the Blue force, dependent upon the specified resolution, scenario, and data base.

The ideas in this chapter are primarily the author's and are tentative, not having yet been modeled. More work and research is required to validate the concepts discussed herein. The organizations, prioritizations, and tactics are for purposes of illustration only.

A. BLUE MISSIONS AND ORGANIZATIONS

The brigade level FA model should simulate the coordination of all the maneuver commander's fire support assets. Fire support elements in the brigade area that should be modeled are:

1 - 81mm mortar

2 - 4.2 inch mortar

3 - 155mm howitzer

4 - 203mm howitzer

5 - General support rocket system (GSRS)

6 - A-10 aircraft (Air Force close air support aircraft)

7 - Advanced attack helicopter (AAH)

A 155mm M109A1 battalion (six 4-gun firing units) would be in direct support of the Blue maneuver brigade. Each battalion task force within the brigade would have a battalion fire support officer (Bn FSO) and three fire support teams (FIST). Each FIST would consist of either one forward observer (armor configuration) or three forward observers (mechanized infantry configuration). One battalion (three 4-gun firing units) of reinforcing artillery would bepresent in the brigade and respond to requests for additional fires from the DS battalion. A battalion of 203mm/GSRS artillery would be in the brigade area, with a tactical mission of general support reinforcing (GSR). This type of mission requires the FA unit to furnish FA fires to the division as a whole and to reinforce the fires of the DS battalion as a second priority. Each battalion task force would be equipped with the appropriate number of 81mm and 4.2 inch mortars, dependent upon their configuration. Close air support assets would be requested through the tactical air control parties (TACP) located at battalion headquarters and at brigade headquarters. The division aviation assets would be requested through the appropriate battalion or brigade level operations officer.

B. RESOURCE ALLOCATION AND THE COUNTERFIRE MISSION

Counterfire is the attack of enemy indirect weapon systems, including indirect fire weapons and target acquisition devices. Counterfire targets would normally be attacked using division (or higher level) assets, such as the GSR battalion located in the brigade area of responsibility. These assets would fire almost exclusively at counterfire targets. The following priority scheme is suggested to be used for counterfire targets:

1 - Nuclear delivery capable units.

2 - Target acquisition units.

3 - Command and control.

4 - Rocket/missile artillery.

5 - Cannon artillery.

6 - Air defense units.

7 - Mortars.

8 - General.

Attacking mortars would be a responsibility of the direct support and reinforcing artillery battalions. The maneuver commander would decide if and when to use the assets within the brigade for this counterfire function.

The modeling of the counterfire functions for the division artillery should be considered separately from the fires of the brigade. Using the simplified network concept from the FA battalion model, the GSR battalion (plus any other battalions) could easily be modeled, provided that a priority scheme is developed to place targets in the waiting queue.

Each Blue fire unit would be designated either a brigade asset or a division artillery asset at the beginning of the battle, based on the scenario and the artillery organization. Those assets that are designated counterfire would only fire counterfires, while those assets that are a brigade asset would fire in support of the brigade using the brigade's priority scheme for targets. The resource allocation of these units could either be fixed or allowed to change dynamically during the course of the battle. For example, if the brigade is withdrawing or in a counterattack, the allocation could be changed.

C. TARGET ACQUISITION

The primary target acquisition assets for the FA brigade model would be the forward observers, the AN/TPQ 36 radar (mortar locating radar), the AN/TPQ 37 radar (artillery locating radar), the airborne standoff target acquisition system (SOTAS), and aerial observers (AOs) from the division

artillery aviation section. The information gathered by each of these sources could generate a target to be evaluated for attack, either by the brigade fire support coordination system or the division artillery counterfire system. Each target so generated would have the following characteristics:

1 - Time target was detected.

2 - Target's movement status: stationary or moving.

3 - Type of mission (categories are the same as described for the brigade's target priority list).

4 - Location accuracy in terms of CEP (circular error probable).

5 - Preempt status - Is this mission important enough to take an asset away from a mission in progress?

6 - Target type (XM1, T72, 155mm battery, etc.).

The radars would be modeled as the primary target acquisition capability for those fire support assets responsible for countermortar and counterbattery fires. Since the AN/TPQ 36 radar is designed to locate mortars, a data link between the AN/TPQ 36 radar and the DS artillery battalion TACFIRE would be established. A similar data link would be established between the AN/TPQ 37 radar and the GSR battalion from the division artillery's assets.

Sound and flash bases would be located in the forward areas of the brigade sector. Their primary responsibility would be to cue the radars (tell them when to turn on). The radars would normally be in a standby mode, to lower their vulnerability to enemy acquisition. The sequence of actions for the sound and flash bases and the radars would be:

1 - Sound or flash base senses that the enemy fired.

2 - Sound or flash base communicates the information to the radar.

3 - The radar turns itself on.

4 - The radar detects the enemy round.

5 - The radar determines the location of the firing unit.

6 - The radar transmits the target information to the appropriate artillery unit.

Actions 1,2,3,4, and 6 would occur based on probability distributions. Action 5 would generate an error in X and Y coordinates that could be applied to the known firing unit location to determine an apparent target location.

The radar would have to be sited considering line of sight, terrain masking, and the technical limitations of the radar being simulated. The model would check the X-Y plane of the radar's detection envelope for the actual enemy firing unit locations. If the firing unit was inside that X-Y plane, the radar would be capable of detecting the round that had been fired.

The SOTAS device would be installed in an aircraft and flown along the brigade front. It would be concerned with detecting moving targets to the rear of the enemey's front line battalions. The air movement module would move the aircraft along a predesignated flight path. The targets acquired and the location error of each would be a function of the primary azimuth of the device and a probability function that would model the SOTAS detection rate. Data concerning any target that was acquired would be transmitted to the maneuver brigade S-2. After a specified delay, the brigade FSO would get the information and determine whether and how to engage the target.

The aerial observers would be modeled in a way similar to the SOTAS, except that an airborne human observer detection routine would be used. The information on the targets that were acquired would be sent directly to the DS battalion FDC.

D. FIRE SUPPORT COORDINATION

Fire support planning and coordination are accomplished by the maneuver commander and the fire support coordination officer at battalion and brigade level.

It can be anticipated that the number of targets acquired and available to be serviced by brigade assets would exceed the brigade's target servicing capability. The maneuver commander must establish priorities for his fire support assets to follow. An example of prioritization would be:

MISSION TYPE	PRIORITY
Direct fire (suppression)	1
Air defense suppression	2
Counterfire (mortars)	3
Smoke	4
Command and control	5
FASCAM	6
CLGP	7
Direct fire (destruction)	8
Rear echelon - supply	9
Rear echelon-assembly area	10

Each target generated by a forward observer is sent to the FIST chief and monitored by the battalion FSO. The FIST chief checks whether the fire support assets available at his level (81mm mortars, 4.2 inch mortars, and direct support artillery) are capable of furnishing effective fire support. The FIST chief checks his list of assets for the most effective means of attacking the target. If the fire support asset is available and the target is within range, the FIST chief sends the target data to that fire support asset. If the most effective asset is not available, the FIST chief searches for the next best asset. If the FIST chief does

not have enough assets available at his level, he requests additional support from the brigade FSO through the Bn FSO. If the target has an immediate priority, it preempts a current mission. The brigade FSO goes through a similar process of matching up his resources with his priority targets. If all assets are busy at the brigade level, the target is placed in a queue, to wait until an appropriate fire support asset becomes available. When the fire support asset becomes available, the FSCOORD at the appropriate level checks for the highest priority target waiting to be serviced. He selects the highest priority target that can be fired, allowing under utilization of the asset in exchange for getting the highest priority target serviced. These decisions depend upon a number of factors and current constraints generated in the scenaria used and the "commander's guidance".

This coordination and decision making function could be simulated through the use of a fire support coordination array (FSCOORD array) of priorities. Based upon military judgement for the situation and on the scenario modeled, a rank order of effectiveness and priority could be devised to facilitate the FSCOORD planning and decisions. A sample array is shown in Table I. As an example of the array's use, consider a mission type of air defense suppression (mission type-2). The array indicates that the best asset is AAH (the lowest number in row two). If the AAH asset was available, the FIST chief would request it through the battalion operations officer. If the AAH was not available, 155mm cannon artillery would be requested. If no appropriate asset was available, the target would be placed in a queue, waiting until an asset became available. When an asset became available, it would take the target with the highest priority as long as the asset was capable of servicing that target.

TABLE I

Fire Support Coordination Array

MISSION TYPE	ASSET TYPE					
	81mm	4.2 in	155	203	A-10	AAH
1	2	1	3	4	5	-
2	-	-	2	3	-	1
3	2	1	3	4	-	5
4	2	1	3	4	5	-
5	-	-	1	2	3	-
6	2	1	3	-	-	-
7	-	-	1	-	-	-
8	-	-	1	-	-	-
9	-	-	1	2	3	-
10	-	-	1	2	3	4

A number of logical rules would have to be formulated to ensure that appropriate decisions were simulated for the FIST chief/Bn FSO and the Bde FSO. The following set of rules if for purposes of illustration only:

1 - If requesting an A-10 asset, enemy air defense would have to be suppressed prior to committing the A-10's (AAH or other asset).

2 - If the target was a moving target, it would be cancelled if more than three minutes had elapsed since the position had been reported.

3 - If the target was stationary, it would be cancelled if more than 20 minutes had elapsed since its position had been reported (this time would be a function of target type).

4 - Targets with a large potential error in location might not be engaged, depending upon the target type (a soft, stationary, important target like a Red division command post would be attacked even if the error in location is potentially large).

5 - Target must be within range of asset.

E. MOVING FIRING UNITS

The Field Artillery moves only to continue the mission or to increase survivability. New locations for artillery units necessary to support anticipated maneuver force movements would be preplanned. When a specified event occurs (e.g. the battalion task force moves to alternate positions or the brigade counterattacks), the artillery will move to the preplanned positions, ensuring that the movement of firing units does not interfere with providing continuous fire support. The duration of travel and the time required to be ready to fire would be predetermined insofar as possible. When the firing unit reaches the new position and

occupies it, the firing unit would be available for missions (minus any assets lost enroute to enemy fire).

A similar method would be used to move firing units to alternate positions within the same general vicinity (1-2 kilometers per move). The criteria that would trigger such a move to an alternate position could be incoming fire, a given damage incurred, or a specified number of volleys fired.

F. CONCLUSION

The battalion Field Artillery model was developed to provide the basic FA building block for the FA brigade model and the major extensions needed to construct the brigade model were discussed. Both the FA battalion model and the framework for the FA brigade model have been developed with the goal of providing a high resolution Field Artillery model that will assist the analyst in understanding the relationships and interactions of Field Artillery in the combat environment. It is intended that the current FA model be extended and improved concurrently with the STAR model, with the objective of providing a versatile mid-resolution combined arms combat model for a wide variety of military applications.

APPENDIX A

INPUT REQUIREMENTS

General

All Simscript input for the FA model is in free field card input; that is, each data value must be separated by a blank space. This is the only formatting requirement. The Simscript language also has a capability to read in two dimensional arrays, reading in the values row by row (e.g., List(1,1), List 1,2),...). A sample card or cards follows the discussion of each set of input values described below. Table II is a complete list of sample data in the required form.

RN.STREAM, DEBUG, FO.MIN.RANGE, FO.MAX.RANGE

RN.STREAM is an integer variable equal to the selected random number stream. DEBUG is an integer variable that designates the amount of FA output required (1 - no additional data, 1 - each event documented, 5 - each event documented and the array called CLUSTERS printed). FO. MIN.RANGE and FO.MAX.RANGE are integer variables whose values are the FO's minimum and maximum allowable range within which he is allowed to initiate a fire mission. These values could appear on a card as:

5 5 100 3800

which represents the fifth random number stream, a DEBUG value that documents each event and prints the array called CLUSTERS, and FO minimum and maximum opening ranges of 100 meters and 3800 meters, respectively.

BOX.TOLERANCE, MISS.TOLERANCE, FWD.OBS.MSN.TOLERANCE

BOX.TOLERANCE is the length of each side of the 'box' used to cluster vehicles. MISS.TOLERANCE is an integer variable equal to the minimum

TABLE II. Complete list of sample input data.

```
5 5 100 3800
300 300 50
3 3 1 11
3 8 1 5
3 8 4 2 1 4
3 0 3
3 0 50
3 78 79
1 -3550 1200
1 -3000  900
1  150 -4500
4 1 -3500  1200  1000 1 18000 3
4 1 -3500   950  1000 1 18000 3
4 1   200 -4500  1000 2 21000 2
6 0 12000 12000  1000 4 18000 2
6 0 14000 13000  1000 4 18000 2
6 0 11500 12500  1000 4 18000 2
6 0 13000 12000  1000 4 18000 2
6 0 14500 13500  1000 4 18000 2
6 0 10500 12500  1000 4 18000 2
6 0 15500 12000  1000 4 18000 2
6 0 15000 15000  1000 5 21000 .1
    0    0    0    0    0    0  100  300  400
    0    0    0    0    0    0  200  700  900
    0    0    0    0    0    0  600 1000 2000
  100  100  200  100  600  100    0    0    0
  100  100  200  100  200  100    0    0    0
    0    0    0    0    0    0   20   40   80
    0    0    0    0    0    0   20   40  100
    0    0    0    0    0    0    0    0    0
   10   10   80   80  100   80    0    0    0
   20   20   60   60  100  100    0    0    0
    0    0    0    0    0    0  200  400  500
    0    0    0    0    0    0  400 1000 1200
    0    0    0    0    0    0 1000 3000 4000
  400  400  400  300 1000  200    0    0    0
  300  300  200  200  400  200    0    0    0
    0    0    0    0    0    0   50  100  200
    0    0    0    0    0    0   40  100  200
    0    0    0    0    0    0    0    0    0
   50   50  100  100  150  100    0    0    0
   50   50  100  100  130  150    0    0    0
    0    0    0    0    0    0  300  500  600
    0    0    0    0    0    0  400  900 1000
    0    0    0    0    0    0  800 2000 3000
  300  300  300  300  700  100    0    0    0
  200  200  100  100  300  100    0    0    0
    0    0    0    0    0    0   30   60  100
    0    0    0    0    0    0   40   80  150
    0    0    0    0    0    0    0    0    0
   50   50   80   80  100  100    0    0    0
   40   40  100  100  150  150    0    0    0
```

Table II (continued)

350	350	600	600	1200	1200	350	350	1200	1200

25	-75
-25	-25
25	25
-25	75

25	-75
-25	-25
25	25
-25	75

40	-120
-40	-40
40	40
-40	120

	6.3		7.6		15
	5.7		6.7		12.7
	15.4		30		11.5
	13.7		19		23
	35		17.2		20.4
30	5	40	30	30	20
60	10	60	40	60	40
90	15	80	50	90	60
40	5	50	30	40	25
70	8	65	40	90	50
100	10	90	50	160	95
80	40	95	50	95	50
200	150	250	180	250	180
250	250	280	350	280	380
40	5	50	30	60	30
80	10	70	40	130	70
120	15	100	50	180	100
100	50	80	40	80	40
150	120	160	100	160	100
120	240	180	230	175	230
120	240	180	230	175	230

1	30	0
2	25	30
2	2	5
2	7.5	12.5
3	25.85	2.25
2	15	20
1	17.5	25
1	10	9

radial distance between the adjusting round and the target, within which the FO is allowed to enter the FFE phase. FWD.OBS.MSN.TOLERANCE is equal to the minimum distance allowed between the two fire missions before the FDC cancels the more recent of the two fire missions. These values could appear on a data card thus:

300 300 50

which represents the size of the box as 300 meters, the tolerance between the adjusting round and the target as 300 meters, and the tolerance between two current missions as 50 meters.

N.FO, N.FDC, N.RADIO, N.BATTERY

The addition of the prefix 'N.' to an entity's name is a system variable that equals the total number of entities of that type. The command, "CREATE EACH 'ENTITY'," creates the number of entities specified in "N.'ENTITY'". N.FO is the number of FOs to be created, N.FDC is the number of FDCs to be created, N.RADIO is the number of radio frequencies to be created, and N.BATTERY is the number of firing units to be created. These values could appear on a card thus:

3 3 1 11

which represents 3 FOs, 3 FDCs, 1 radio frequency, and 11 firing units.

AMT.BLUE.BATTERYS, AMT.RED.BATTERYS, MAX.NUMBER.OF.MISSIONS.PER.FO, AMT.CALIBERS

AMT.BLUE.BATTERYS is the number of Blue firing units created. AMT.RED.BATTERYS is the number of Red firing units created. MAX.NUMBER.OF.MISSIONS.PER.FO is the maximum number of fire missions that any FO is allowed to process simultaneously. AMT.CALIBERS is the number of different artillery weapon systems. These values could appear on a data card thus:

3 8 1 5

which represents 3 Blue firing units, 8 Red firing units, the maximum missions per FO as 1, and 5 different artillery weapon systems.

NO.RANGE.BANDS, AMT.FA.TIME.DELTAS, LARGEST.NUM.WPNS, AMT.AMMO.TYPES, AMT.MRL, AMT.FFE.VOLLEYS

NO.RANGE.BANDS is the number of range bands used in array RANGE. BANDS. AMT.FA.TIME.DELTAS is the number of time parameters used in array FA.TIME.DELTAS. LARGEST.NUM.WEAPONS is the maximum number of weapons in any Blue firing unit. AMT.AMMO.TYPES is the number of different types of ammunition. AMT.MRL is the number of MRL batteries. AMT.FFE.VOLLEYS is the number of the volleys to be fired in the FFE phase. These values could appear on the data card thus:

3 8 4 2 1 4

which represents 3 range bands, 8 different time parapmeters, 4 weapons, 2 types of ammunition, 1 MRL battery, and 4 volleys in the FFE phase.

TGT.ACQ.ERROR

TGT.ACQ.ERROR is a two dimentionsl array containing the error parameters of the error distribution for the FO's observation device, different for each device (measured in meters). The first subscript indicates the type of observation device (1 - laser, 2 - binoculars) and the second subscript indicates the parameters that characterize the distribution. If the value in the first column equals 1, the error is deterministic with a value equal to the second column. If the value in the first column is equal to 2, the distribution is uniform, with endpoints in the second and third columns. If the value in the first column is equal to 3, the error is from a normal distribution, with a mean equal to the value in the second column, and a standard deviation equal to the

value in the third column.

The data might appear on two cards thus:

3	0	2.5
3	0	50

which represents the error as resulting from the use of a laser as from a Normal(0,2.5) distribution and the error resulting from the use of binoculars as Normal(0,50).

FO.VEHICLE

FO.VEHICLE is a one dimensional array containing the names of the vehicles in which the FOs are riding. The array might appear on a data card thus:

3 78 79

which indicates that the FOs are riding in vehicles 3, 78, and 79.

Attributes

A number of attributes of permanent entities must be input on data cards. In the discussion that follows, real variables will be denoted by an R in parentheses following the attribute name. Real attributes may be input with or without decimal points. Integer variables will be denoted by an I in parentheses following the attribute name.

FDC

COLOR.2 (I) - A value of 1 indicates a Blue unit. A value of 0 denotes a Red unit.

X.CUR2 (R) - The FDC's X coordinate on the battlefield.

Y.CUR2 (R) - The FDC's Y coordinate on the battlefield.

The data might appear thus:

1 -3550 1200

1	-3000	900
1	150	-4500

(Negative coordinates indicate the the FDC is outside the 10x10 kilometer box used for the ground war battlefield.)

BATTERY

NUM.GUNS (I) - Number of weapons.

COLOR1 (I) - A value of 1 indicates a Blue firing unit. A value of 0 indicates a Red firing unit.

X.CUR1 (R) - The firing unit's X coordinate on the battlefield.

Y.CUR1 (R) - The firing unit's Y coordinate on the battlefield.

NUM.DPICM.LEFT (I) - The basic load of DPICM rounds for each firing unit.

CALIBER (I) - The type of artillery weapon system:

1	155mm
2	203mm
3	GSRS
4	152mm
5	122mm MRL

MAX.RANGE (I) - The maximum range of the firing unit.

RATE.OF.FIRE (R) - The rate of fire of the firing unit (in rounds per minute).

The data might appear on cards as shown in Table III.

ARTY.PK.TABLE, LETHAL.RADIUS.ARRAY

The appropriate lethal area for the damage level, ammunition type used, type artillery weapon system fired, and type target fired upon is read in and converted to probabilities of damage (ARTY.PK.TABLE)

TABLE III. Sample data input for attributes of each battery.

4	1	-3500	1200	1000	1	18000	3
4	1	-3500	950	1000	1	18000	3
4	1	200	-4500	1000	2	21000	2
6	0	12000	12000	1000	4	18000	2
6	0	14000	13000	1000	4	18000	2
6	0	11500	12500	1000	4	18000	2
6	0	13000	12000	1000	4	18000	2
6	0	14500	13500	1000	4	18000	2
6	0	10500	12500	1000	4	18000	2
6	0	15500	12000	1000	4	18000	2
6	0	15000	15000	1000	5	21000	.1

and lethal radius (LETHAL.RADIUS.TABLE). The data is in the form of a 4-dimensional array. The first subscript indicates the damage level, the second subscript indicates the ammunition type, the third subscript indicates the weapon type, and the fourth subscript indicates the target type. The four subscripts have the following values:

Damage level:

1 - k-kill

2 - f-kill

3 - m-kill

Ammunition type:

1 - DPICM

2 - HE

Artillery weapon type:

1 - 155mm

2 - 203mm

3 - GSRS

4 - 152mm

5 - 122mm MRL

Target type:

1 - XM1 105mm main gun

2 - XM1 120mm main gun

3 - IFV

4 - ITV

5 - DIVAD

6 - DRAGON

7 - T72

8 - BMP

9 - ZSU

A typical data set might appear as follows:

0	0	0	0	0	0	100	300	400
0	0	0	0	0	0	200	700	900
0	0	0	0	0	0	600	1000	2000
100	100	200	100	600	100	0	0	0
100	100	200	100	200	100	0	0	0

This is a subset of the entire four dimensional array, with the first subscript equal to 1 (k-kill) and the second subscript equal to 1 (DPICM). In regards to this input array the rows of the array represent the weapon type and the columns of the array represent the different types of targets that are being assessed for damage. As an example of the use of this array, consider the data set above. The lethal area (k-kill) associated with the 155mm round (DPICM) for a T72 (target type 7) would be 100 meters.

TRAVEL.TIME.ARRAY

TRAVEL.TIME.ARRAY is a two dimensional array that contains the average velocity at two-thirds maximum range for each artillery system/ammunition type. The first subscript indicates the type of ammunition and the second subscript indicates the artillery weapon type.

Subscript values:

Ammunition type:

1 - DPICM

2 - HE

Artillery weapon type:

1 - 155mm

2 - 203mm

3 - GSRS

4 - 152mm

5 - 122mm MRL

These values could appear on data cards thus:

350 350 600 600 1200 1200 350 350 1200 1200

where the first entry represents the velocity of 350 meters per second (m/s) for DPICM/155mm, the second entry represents a velocity of 350 m/s for HE/155mm, and the third entry represents the velocity of 600 m/s for DPICM/203mm.

DISPLACEMENT

DISPLACEMENT is a real 3-dimensional array containing the displacement of each weapon from the center of the firing unit (Blue firing units only). The measurement is done in the battlefield coordinate ysstem with the firing unit pointing East (measured from the horizontal plane parallel to the X axis). The first subscript indicates the weapon number (1 for the first weapon, 2 for the second weapon, and so on), the second subscript indicates the type of coordinate, and the third subscript indicates the number of the firing unit.

Subscript values:

Weapon number:

The number of the weapon in the firing unit.

Type of coordinate:

1 - The X coordinate.

2 - The Y coordinate.

Number of firing unit:

The order in which the firing unit was created (e.g., the first firing unit created is 1, the second firing unit created is 2, etc.).

The data for the first firing unit (the third subscript is equal to 1) might appear on data cards thus:

25	-75
-25	-25
25	25
-25	75

In this example, the rows indicate the weapon number and the columns represent the coordinate type. As an example, the X coordinate displacement for the second weapon (first firing unit) would be -25 meters from the center of the first firing unit and the Y coordinate displacement for the second weapon (first firing unit) would be -25 meters from the center of the first firing unit.

RANGE.BANDS

RANGE.BANDS is a two dimensional array containing the range bands of the piecewise linear approximations of the impact point dispersion curves. In the array, the first subcript indicates the breakpoint in kilometers from the weapon and the second subscript indicates the type of artillery weapon system.

Subscript values:

Breakpoint:

The appropriate breakpoint (endpoint) of the linear approximation.

Artillery weapon type:

1 - 155mm

2 - 203mm

3 - GSRS

4 - 152mm

5 - 122mm MRL

The data might appear on cards thus:

6.3	7.6	15	5.5	6.7
12.7	15.4	30	11.5	13.7
19	23	35	17.2	20.4

The rows indicate the breakpoints and the columns indicate the different types of artillery weapons. As an example, to determine the range bands for the 203mm (second column): the first range band starts at 7.6 kilometers (the first breakpoint), the second range band starts at 15.4 kilometers (the second breakpoint), and the second range band stops at 23 kilometers (the third breakpoint).

SIGMA.DPICM

SIGMA.DPICT is a three dimensional array containing the standard deviations for the normal distribution that characterizes the dispersion of round/MPI impact (DPICM only), in meters. The first subscript indicates the range band, the second subscript indicates the type of standard deviation, and the third subscript indicates the type of artillery weapon. Subscript values:

Range band:

The representative range band.

Type of standard deviation:

1 - Standard deviation (range) for precision accuracy (round to round variation).

2 - Standard deviation (deflection) for precision accuracy.

3 - Standard deviation (range) for observer adjusted (MPI).

4 - Standard deviation (deflection) for observer adjusted (MPI for a volley).

5 - Standard deviation (range) for MET + VE (meteorological and velocity error) (unobserved fire that used registration

firings that corrected for meteorological conditions).

6 - Standard deviation (deflection) for MET + VE.

Artillery weapon type:

1 - 155mm

2 - 203mm

3 - GSRS

4 - 152mm

5 - 122mm MRL

The two dimensional subset array that applies to the 155mm (the third subscript is equal to 1) might appear on cards thus:

30	5	40	30	30	20
60	10	60	40	60	40
90	15	80	50	90	60

where the row indicates the range band and the column is equal to the values 1 to 6. As an example, the standard deviation of range for precision accuracy (155mm) is 30 meters at the first breakpoint of the range band, the standard deviation is 60 meters at the second breakpoint and the value is 90 meters at the last breakpoint (end of the second range band).

FA.TIME.DELTAS

FA.TIME.DELTAS is a two dimensional array containing the parameters of the distributions that characterize the time required to accomplish specified tasks (measured in seconds). The first subscript indicates the task and the second subscript indicates the parameter of the distribution.

Task:

1 - Update cluster.

2 - Time required for FO to calculate his mission and code his mission into his message device.

3 - Time for the FO to communicate fire request to FDC.

4 - Processing time in the FDC.

5 - Time required in the firing battery to fire.

6 - Time for the FO - subsequent adjusting round.

7 - Time required for the firing unit - adjusting round.

8 - Time required for FO to end the mission after last volley impacts.

If the value in the first column equals 1, the error is deterministic with a value equal to that in the second column. If the value in the first column is 2, the distribution is uniform with end points in the second and third columns. If the value in the first column is equal to 3, the distribution is normal, with a mean equal to the value in column two, and a standard deviation equal to the value in the third column.

The data might appear on cards thus:

1	30	0
2	25	30
2	2	5
2	7.5	12.5
3	25.85	2.25
2	15	20
1	17.5	25
1	10	9

As an example, the time required for the FO to transmit his message (row 3) is characterized by a uniform distribution, with end points 2 and 5.

BIBLIOGRAPHY

1. Wallace, W. S. and Hagewood, E. G., Simulation of Tactical Alternative Responses (STAR), M.S. Thesis, Naval Postgraduate School, Monterey, December 1978.

2. Kelleher, E. P., Simulation of the Tactical Employment of Field Artillery, M.S. Thesis, Naval Postgraduate School, Monterey, December 1977.

3. Department of the Army, FM-6-20, Fire Support in the Combined Arms Operations, May 1977.

4. U.S. Army Material Development and Readiness Command, DARCOM-P 706-101, Engineering Design Handbook, Army Weapon Systems Analysis, Part One, November 1977.

5. Stockfisch, J. A., Models, Data, and War: A Critique of the Study of Conventional Forces, RAND, March 1975.

6. Kiviat, P. J., Villanueva, R., and Markowitz, H. M., Simscript II.5 Programming Language, 2nd Edition, Consolidated Analysis Center, Inc., 1973.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 55 Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
4.	Professor J. K. Hartman, Code 55Hh (Thesis Advisor) Department of Operations Research Naval Postgraduate School Monterey, California 93940	5
5.	LTC Edward P. Kelleher, Code 55Ka (Thesis Advisor) Department of Operations Research Naval Postgraduate School Monterey, California 93940	5
6.	Professor S. A. Parry, Code 55Py Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
7.	CPT S. G. Starner 388A Rickets Road Monterey, California 93940	1
8.	Office of the Commanding General U.S. Army TRADOC Attn: General Donn A. Starry Ft. Monroe, Virginia 23651	1
9.	Headquarters U.S. Army Training & Doctrine Command Attn: ATCG-T (Col. Ed Scribner) Ft. Monroe, Virginia 23651	1

10. Headquarters 1
U.S. Army Training & Doctrine Command
Attn: Director,Analysis Directorate -
Combat Developments (Col. Tony Pokorney)
Ft. Monroe, Virginia 23651

11. Headquarters 1
U.S. Army Training & Doctrine Command
Attn: Director, Maneuver Directorate -
Combat Developments (Col. Fred Franks)
Ft. Monroe, Virginia 23651

12. Lt. General J. R. Thurman 1
Commanding General
U.S. Army Combined Arms Center
Ft. Leavenworth, Kansas 66027

13. Director 1
Combined Arms Combat Development Activity
Attn: Col Reed
Ft. Leavenworth, Kansas 66027

14. Dr. Wilbur Payne, Director 1
U.S. Army TRADOC Systems Analysis Activity
White Sands Missile Range, New Mexico 88002

15. Director 1
Armored Combat Vehicle Technology Program
Attn: Col. Fitzmorris
U.S. Army Armor Center
Ft. Knox, Kentucky 40121

16. Director 1
Studies Division - Combat Developments
Attn: Col. Burnett
U.S. Army Armor Center
Ft. Knox, Kentucky 40121

17. Col. Frank Day 1
TRADOC Systems Manager - XM1
U.S. Army Armor Center
Ft. Knox, Kentucky 40121

18. MG Dick Lawrence 1
Tank Forces Management Office
Room 1A-871, The Pentagon
Washington, D.C. 20310

19. Mr. David Hardison 1
Deputy Undersecretary of the Army
(Operations Research)
Department of the Army, The Pentagon
Washington, D.C. 20310

20. Director 1
U.S. Army Material Systems Analysis Activity
Attn: Mr. Will Brooks
Aberdeen Proving Grounds, Maryland 21005

21. Command and General Staff College 1
Attn: Educational Advisor
Room 123, Bell Hall
Ft. Leavenworth, Kansas 66027

22. Headquarters, Department of the Army 1
Office of the Deputy Chief of Staff
for Operations and Plans
Attn: DAMO-2D
Washington, D.C. 20310

23. Commandant 1
U.S. Army Field Artillery School
Attn: ATSF-MBT
Fort Sill, Oklahoma 73503

24. Commandant 1
U.S. Army Field Artillery School
Attn: ATSF-CD
Fort Sill, Oklahoma 73503

25. Royal Armament Research and Development Establishment
MOD(PE), Attn: S/MA4 - Mr. A. D. Cox 1
Fort Halstead
Seven Oaks
Kent
TN14 7BP
United Kingdom